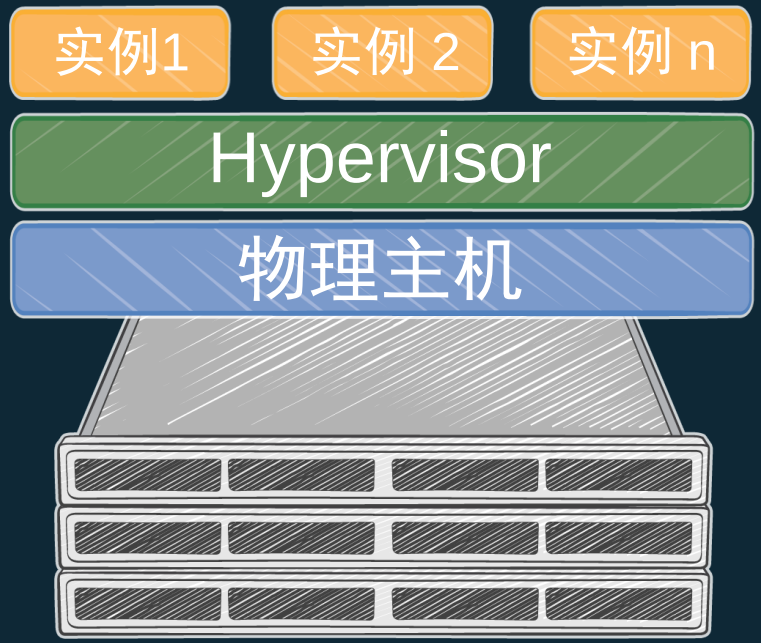
Amazon EC2实例深入理解和性能优化

丁成银， AWS解决方案架构师
Chandler Ding, Solution Architect, Amazon Web Services

目录

- Understanding the factors that going into **choosing an EC2 instance**
- Defining **system performance** and how it is characterized for different workloads
- How Amazon **EC2 instances deliver performance** while providing flexibility and agility
- How to **make the most** of your EC2 instance experience through the lens of several instance types

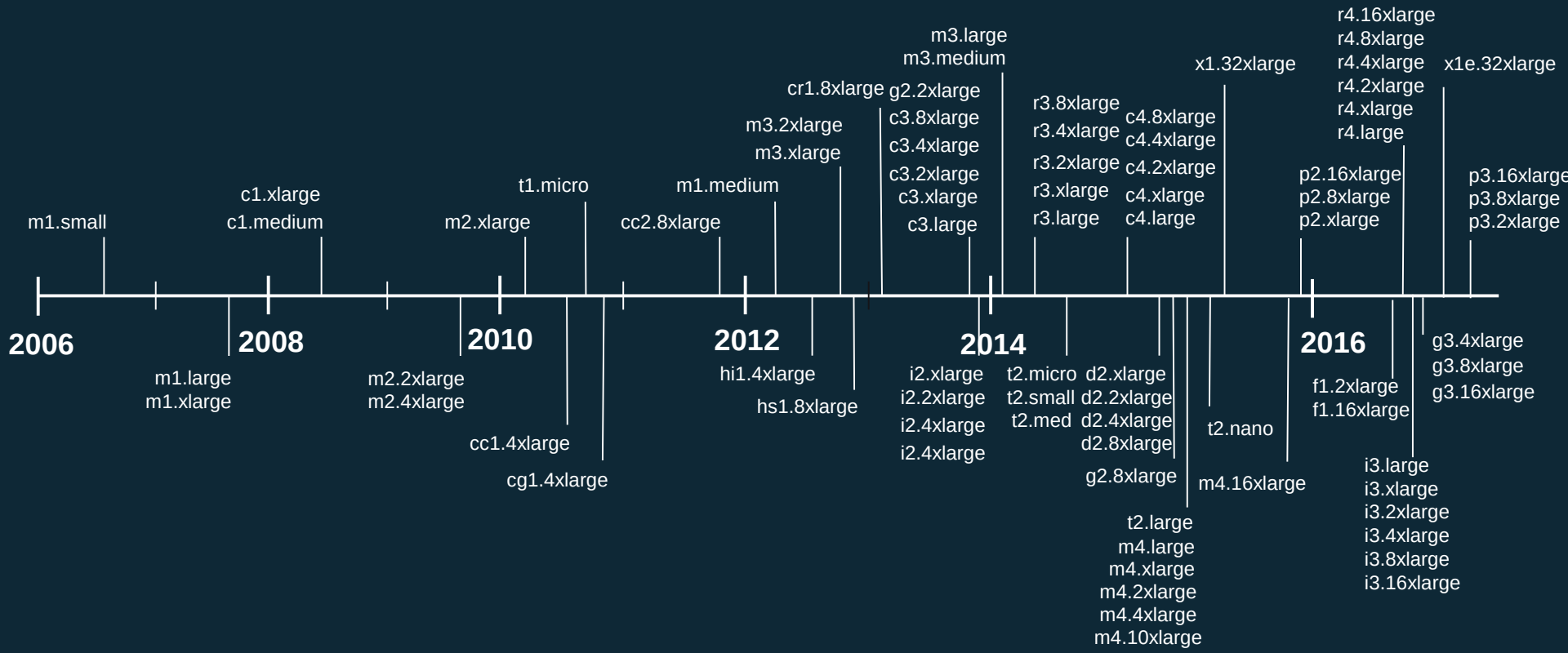
Amazon EC2 Instances



很久以前

- 2006年8月上线
- 第一代EC2实例： M1实例

EC2实例时间线


2006
m1.small
2008
c1.xlarge
c1.medium
m1.large
m1.xlarge
2010
m2.xlarge
t1.micro
m2.2xlarge
m2.4xlarge
cc1.4xlarge
cg1.4xlarge
2012
cc2.8xlarge
m1.medium
m3.2xlarge
m3.xlarge
hi1.4xlarge
hs1.8xlarge
2014
cr1.8xlarge
m3.large
m3.medium
g2.2xlarge
c3.8xlarge
c3.4xlarge
c3.2xlarge
c3.xlarge
c3.large
i2.xlarge
i2.2xlarge
i2.4xlarge
i2.4xlarge
r3.8xlarge
r3.4xlarge
r3.2xlarge
r3.xlarge
r3.large
c4.8xlarge
c4.4xlarge
c4.2xlarge
c4.xlarge
c4.large
t2.micro
t2.small
t2.med
d2.xlarge
d2.2xlarge
d2.4xlarge
d2.8xlarge
g2.8xlarge
t2.large
m4.large
m4.xlarge
m4.2xlarge
m4.4xlarge
m4.10xlarge
x1.32xlarge
t2.nano
m4.16xlarge
2016
r4.16xlarge
r4.8xlarge
r4.4xlarge
r4.2xlarge
r4.xlarge
r4.large
p2.16xlarge
p2.8xlarge
p2.xlarge
x1e.32xlarge
p3.16xlarge
p3.8xlarge
p3.2xlarge
f1.2xlarge
f1.16xlarge
g3.4xlarge
g3.8xlarge
g3.16xlarge
i3.large
i3.xlarge
i3.2xlarge
i3.4xlarge
i3.8xlarge
i3.16xlarge

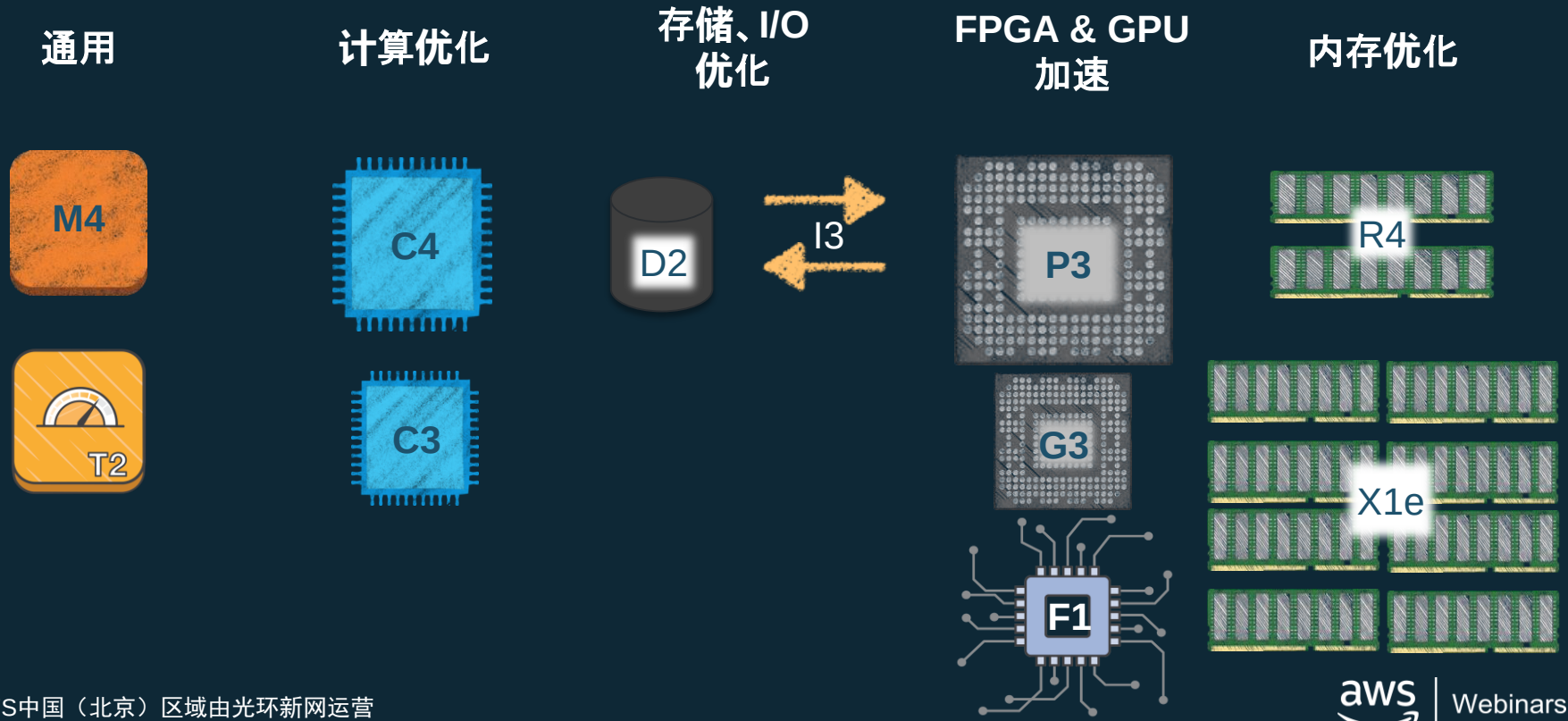
代数

c4.xlarge

家族

大小

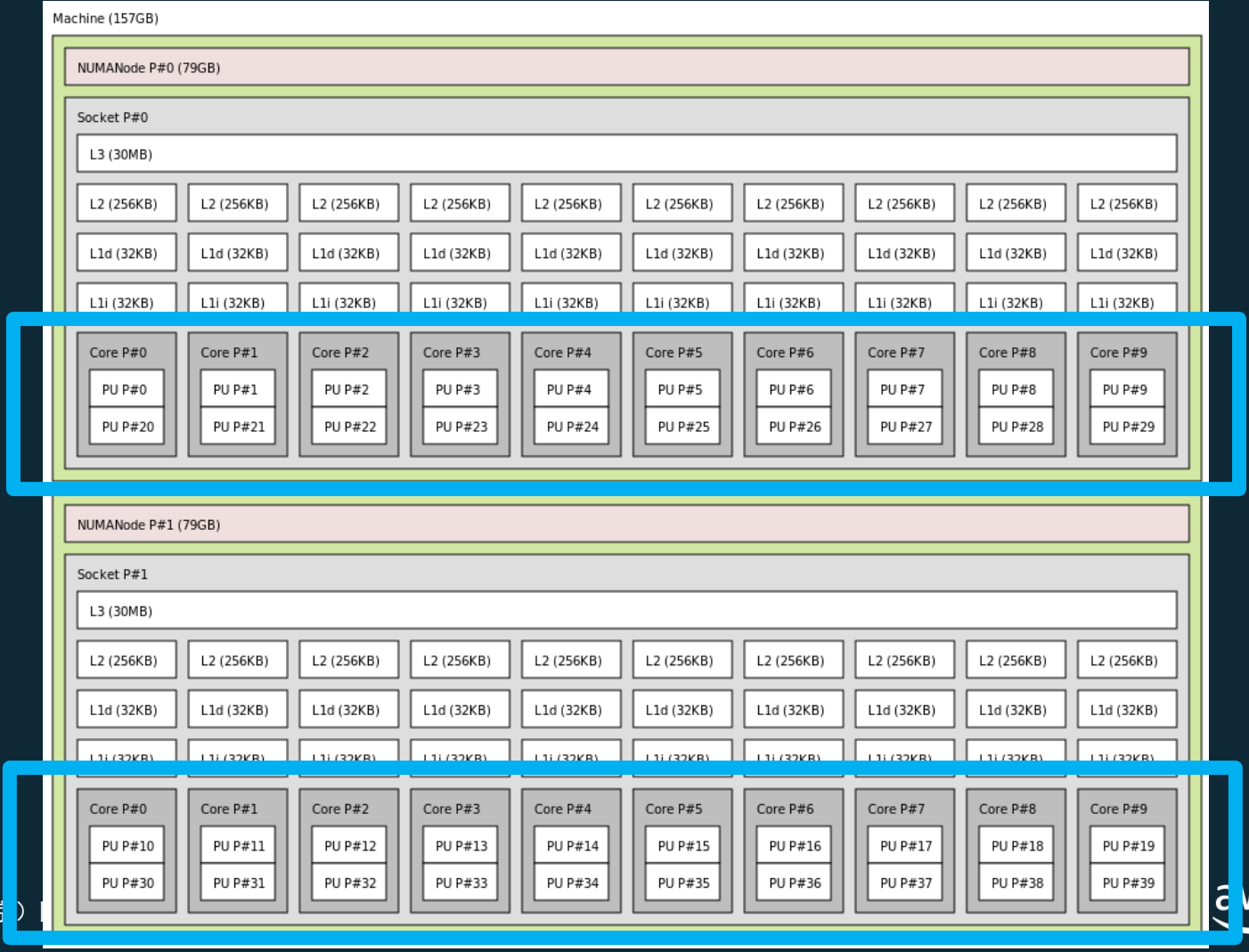
EC2 Instance Families



什么是虚拟CPU(vCPU)

- 一个vCPU对应一个超线程核*
- vCPU数量÷2得到物理CPU核数
- EC2和RDS实例的核数:
  https://aws.amazon.com/ec2/virtualcores/

* T2除外，因为它被设计为共享资源并可突发


Machine (157GB)
NUMANode P#0 (79GB)
Socket P#0
L3 (30MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 PU P#0 PU P#20
Core P#1 PU P#1 PU P#21
Core P#2 PU P#2 PU P#22
Core P#3 PU P#3 PU P#23
Core P#4 PU P#4 PU P#24
Core P#5 PU P#5 PU P#25
Core P#6 PU P#6 PU P#26
Core P#7 PU P#7 PU P#27
Core P#8 PU P#8 PU P#28
Core P#9 PU P#9 PU P#29
NUMANode P#1 (79GB)
Socket P#1
L3 (30MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
Core P#0 PU P#10 PU P#30
Core P#1 PU P#11 PU P#31
Core P#2 PU P#12 PU P#32
Core P#3 PU P#13 PU P#33
Core P#4 PU P#14 PU P#34
Core P#5 PU P#15 PU P#35
Core P#6 PU P#16 PU P#36
Core P#7 PU P#17 PU P#37
Core P#8 PU P#18 PU P#38
Core P#9 PU P#19 PU P#39

为什么以及如何关闭超线程

- 例如某些浮点运算密集的应用

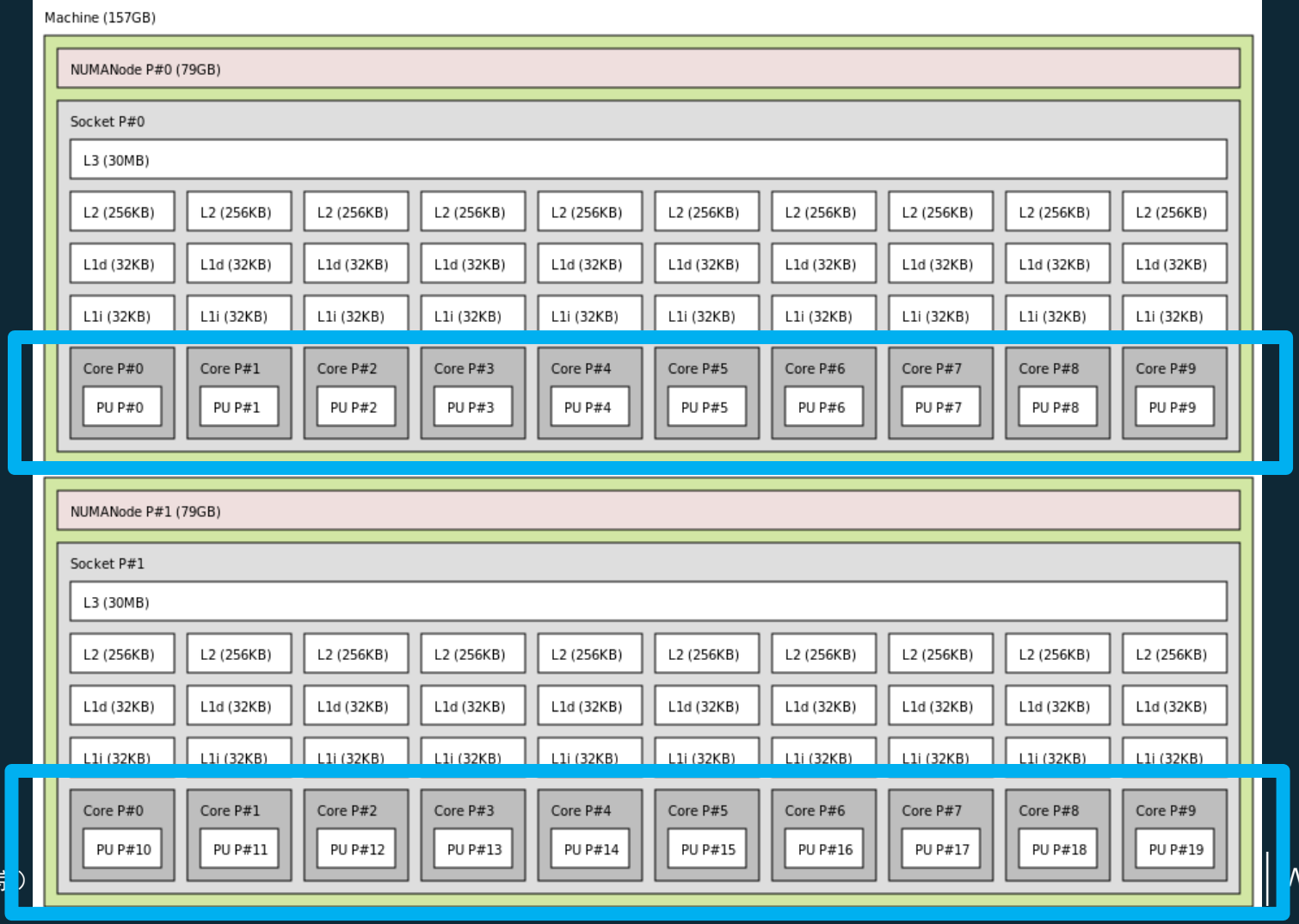
- 运行 ‘lscpu’ 确认拓扑
- 在线关闭“B线程”

```
for i in `seq 64 127`; do
    echo 0 > /sys/devices/system/cpu/cpu${i}/online
done
```

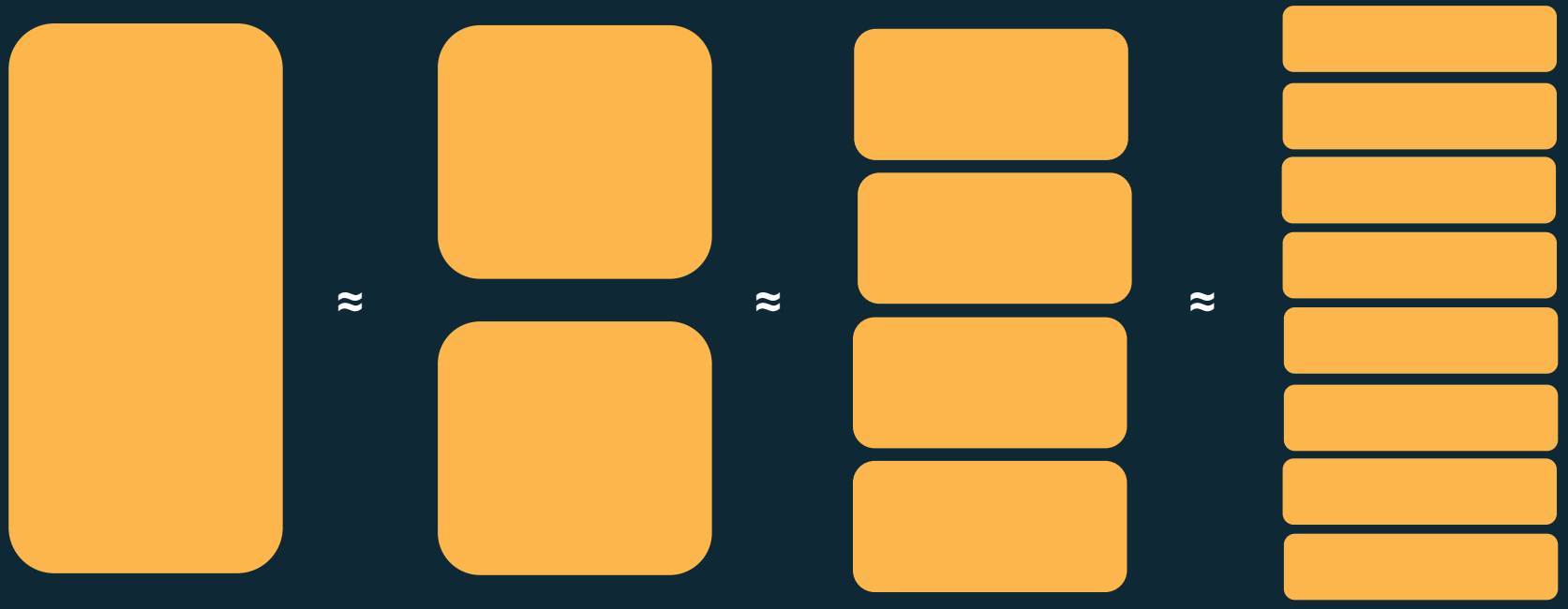
- 修改Linux内核启动参数，只启用前一半vCPU

  `maxcpus=63`

```
[ec2-user@ip-172-31-7-218 ~]$ lscpu
CPU(s):                128
On-line CPU(s) list:   0-127
Thread(s) per core:    2
Core(s) per socket:    16
Socket(s):             4
NUMA node(s):          4
Model name:            Intel(R) Xeon(R) CPU
Hypervisor vendor:     Xen
Virtualization type:   full
NUMA node0 CPU(s):     0-15,64-79
NUMA node1 CPU(s):     16-31,80-95
NUMA node2 CPU(s):     32-47,96-111
NUMA node3 CPU(s):     48-63,112-127
```


Machine (157GB)
NUMANode P#0 (79GB)
Socket P#0
L3 (30MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 PU P#0
Core P#1 PU P#1
Core P#2 PU P#2
Core P#3 PU P#3
Core P#4 PU P#4
Core P#5 PU P#5
Core P#6 PU P#6
Core P#7 PU P#7
Core P#8 PU P#8
Core P#9 PU P#9
NUMANode P#1 (79GB)
Socket P#1
L3 (30MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 PU P#10
Core P#1 PU P#11
Core P#2 PU P#12
Core P#3 PU P#13
Core P#4 PU P#14
Core P#5 PU P#15
Core P#6 PU P#16
Core P#7 PU P#17
Core P#8 PU P#18
Core P#9 PU P#19

Instance sizing



c4.8xlarge | 2 - c4.4xlarge | 4 - c4.2xlarge | 8 - c4.xlarge

资源分配

- 所有分配给实例的资源都是独享的，没有超卖*
  - vCPUs 是独占的
  - 内存是独占的
  - 网络资源通过合理分配来避免“吵闹邻居”的影响
- 想知道每台物理主机能够容纳多少EC2实例? 请参考专用主机 (Dedicated Hosts) 的文档.

* T2除外，因为它被设计为共享资源并可突发

“启动新实例和并行运行测试的过程很简单 …[但是在选择实例类型和规格的时候] 衡量您的应用程序的性能的过程是无可替代的.”

- https://amazonaws-china.com/cn/ec2/instance-types/

关于时钟

- 在EC2实例中提供时钟远比想象的要困难
- gettimeofday(), clock_gettime(), QueryPerformanceCounter()
- 时间戳计数器Time Stamp Counter
  - 用户态可访问的CPU 计数器
  - 需要校准, vDSO
  - 在Sandy Bridge及更新的处理器上是恒常的
- Xen pvclock; 不支持vDSO
- 在最新一代的实例上, 建议使用TSC 作为时钟源

跑个分 - Benchmark

```
#include <sys/time.h>
#include <time.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
time_t start,end;
time (&start);
        for ( int x = 0; x < 100000000; x++ ) {
                float f;
                float g;
                float h;
                f = 123456789.0f;
                g = 123456789.0f;
                h = f * g;
                struct timeval tv;
                gettimeofday(&tv, NULL);
        }
time (&end);
double dif = difftime (end,start);
printf ("Elasped time is %.2lf seconds.\n", dif );
return 0;
}
```

Xen时钟源

```
[centos@ip-192-168-1-77 testbench]$ strace -c ./test
Elasped time is 12.00 seconds.
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 99.99    3.322956           2   2001862           gettimeofday
  0.00    0.000096           6        16           mmap
  0.00    0.000050           5        10           mprotect
  0.00    0.000038           8         5           open
  0.00    0.000026           5         5           fstat
  0.00    0.000025           5         5           close
  0.00    0.000023           6         4           read
  0.00    0.000008           8         1         1 access
  0.00    0.000006           6         1           brk
  0.00    0.000006           6         1           execve
  0.00    0.000005           5         1           arch_prctl
  0.00    0.000000           0         1           munmap
------ ----------- ----------- --------- --------- ----------------
100.00    3.323239              2001912         1 total
```

TSC时钟源

```
[centos@ip-192-168-1-77 testbench]$ strace -c ./test
Elasped time is 2.00 seconds.
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 32.97    0.000121           7        17           mmap
 20.98    0.000077           8        10           mprotect
 11.72    0.000043           9         5           open
 10.08    0.000037           7         5           close
  7.36    0.000027           5         6           fstat
  6.81    0.000025           6         4           read
  2.72    0.000010          10         1           munmap
  2.18    0.000008           8         1         1 access
  1.91    0.000007           7         1           execve
  1.63    0.000006           6         1           brk
  1.63    0.000006           6         1           arch_prctl
  0.00    0.000000           0         1           write
------ ----------- ----------- --------- --------- ----------------
100.00    0.000367                    53         1 total
```

性能小提示: 配置TSC作为系统时钟源

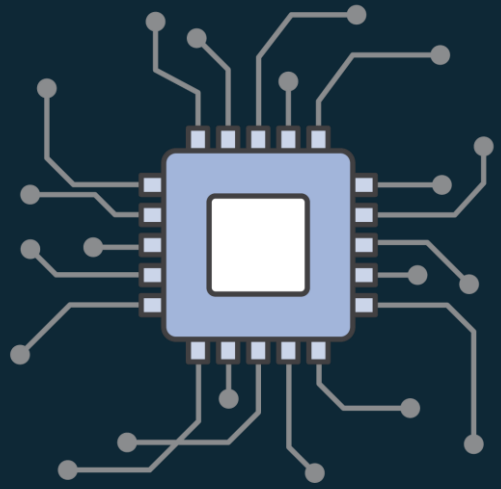
```
# cat /sys/devices/system/cl*/cl*/available_clocksource
xen tsc hpet acpi_pm

# cat /sys/devices/system/cl*/cl*/current_clocksource
xen
```

修改当前时钟源的命令：

```
# echo tsc > /sys/devices/system/cl*/cl*/current_clocksource
```

控制处理器的P-state和C-state

- c4.8xlarge f1.16xlarge p2.16xlarge g3.16xlarge d2.8xlarge i3.8xlarge i3.16xlarge m4.10xlarge m4.16xlarge r4.8xlarge r4.16xlarge x1.16xlarge x1.32xlarge x1e.32xlarge
- 空闲核心进入深层C-State, 工作核心可以睿频到更高主频
- 但是深层C状态的核心需要更长时间来唤醒，可能不适合需要保持稳定的延迟的应用
- 在Grub配置中增加启动参数“intel_idle.max_cstate=1” 来指示CPU只进入浅层C状态



性能小提示：使用AVX2的应用的P-State控制

- 如果应用程序在所有CPU上密集执行Intel高级矢量扩展指令AVX / AVX2
- 密集执行AVX / AVX2指令会产生大量热量，可能会导致CPU自动降频
- 频繁的降频和睿频可能会影响应用的性能的平稳表现
- AVX指令不需要很高的主频也能表现的很好
- 这时可以禁用睿频，让CPU运行在基准频率

```
sudo sh -c "echo 1 > /sys/devices/system/cpu/intel_pstate/no_turbo"
```

参考: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/processor_state_control.html

回顾: T2 实例

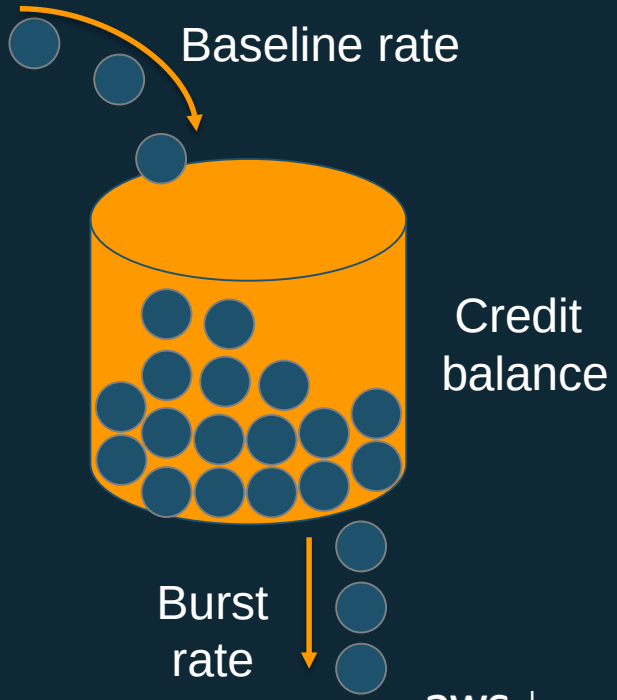
- 最便宜的EC2实例，$0.0065 / 小时 起
- 可突发的性能
- 使用CPU额度来控制

Model	vCPU	Baseline	CPU Credits / Hour	Memory (GiB)	Storage
t2.nano	1	5%	3	.5	EBS Only
t2.micro	1	10%	6	1	EBS Only
t2.small	1	20%	12	2	EBS Only
t2.medium	2	40%**	24	4	EBS Only
t2.large	2	60%**	36	8	EBS Only
t2.xlarge	4	90%	54	16	EBS Only
t2.2xlarge	8	135%	81	32	EBS Only

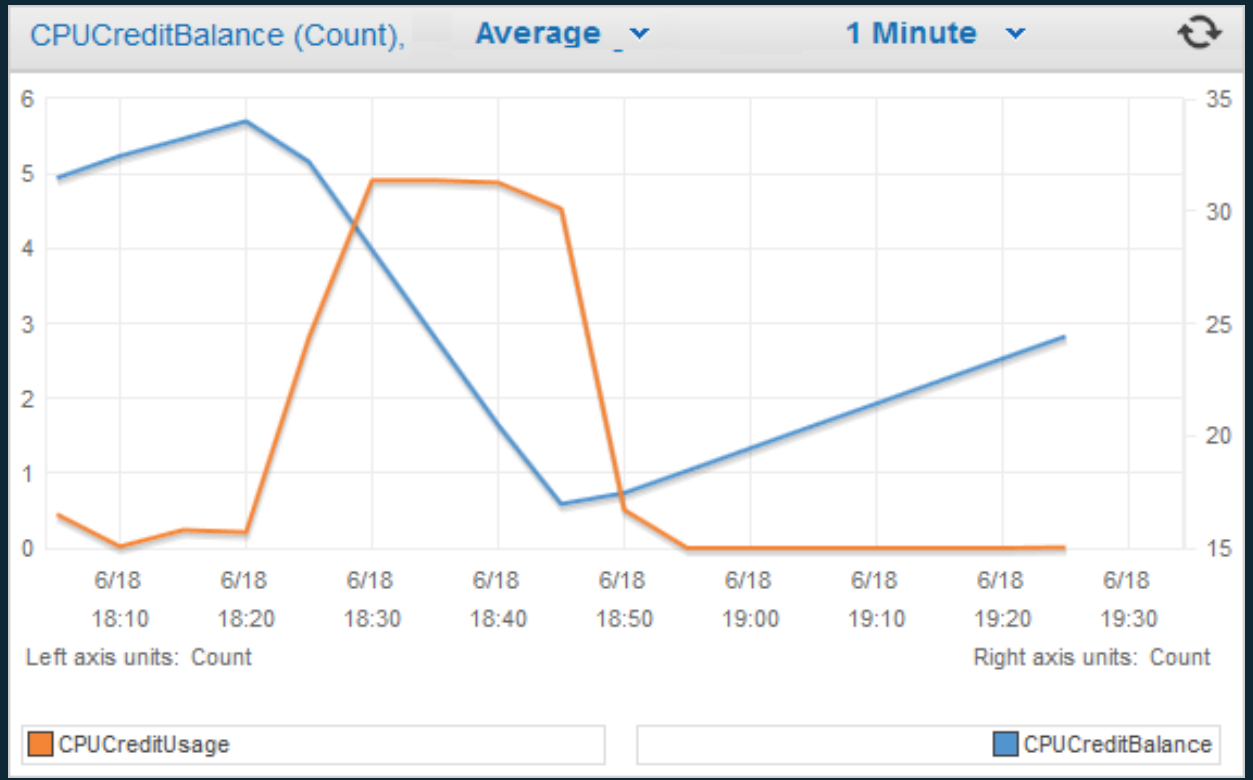
通用，Web服务，开发测试环境，小型数据库

T2的CPU额度原理

- 1个CPU额度可以支持一个CPU核心全速运行1分钟
- 给定规格的T2实例以固定速率获得CPU额度
- 实例占用CPU运行时消耗CPU额度
- 额度最长有效期为24小时



性能提示: 监控CPU额度



回顾: X1e 实例

- 最大4 TB 内存，已经公布了最大16TB内存的路标
- 四插槽128vCPU, Intel E7 处理器

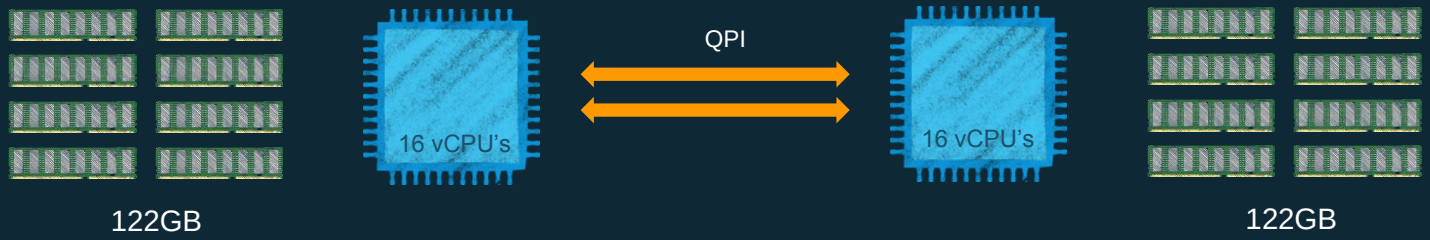
机型	vCPU	内存(GiB)	本地存储	网络带宽
x1.16xlarge	64	976	1x 1920GB SSD	10Gbps
x1.32xlarge	128	1952	2x 1920GB SSD	25Gbps
x1.32xlarge	128	3904	2x 1920GB SSD	25Gbps

适合内存数据库(HANA), 大数据, HPC场景

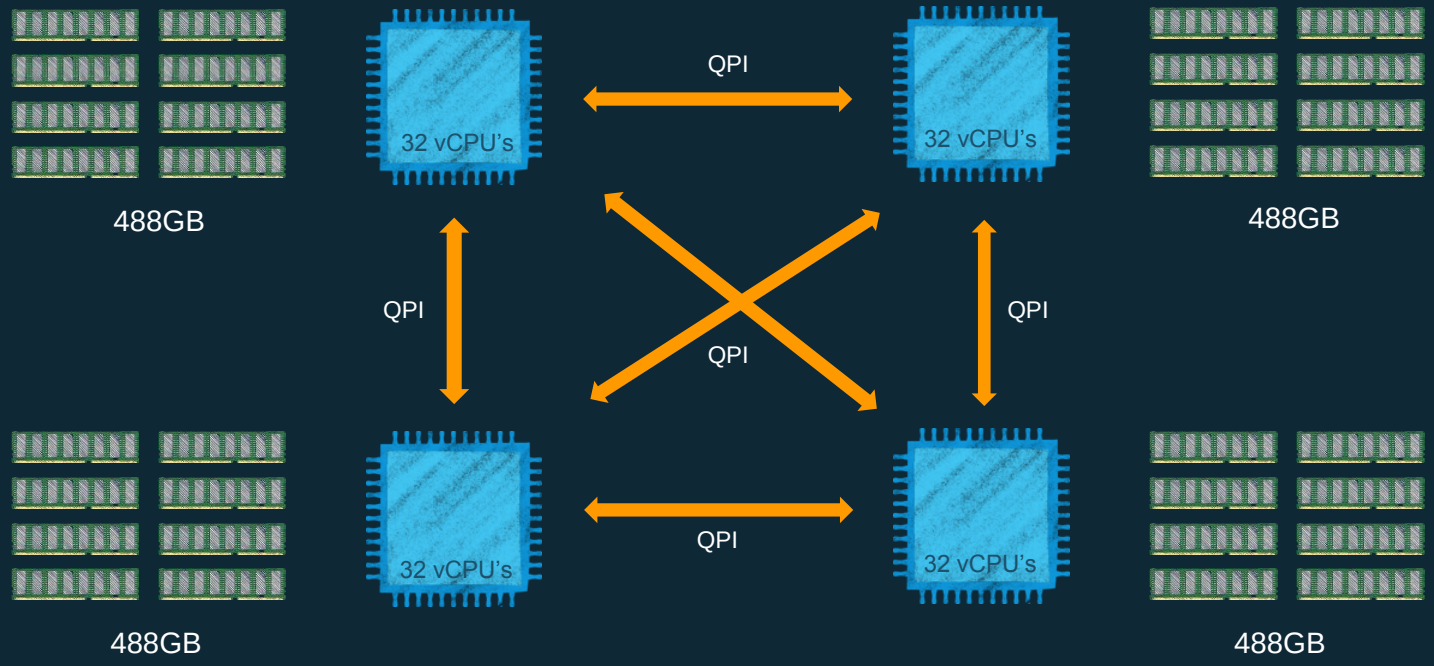
NUMA

- 非一致性访存 Non-uniform memory access
- 多CPU系统中的NUMA架构中，处理器访问它自己的本地内存的速度比非本地内存（内存位于另一个处理器）快一些。
- 访问远端内存的性能取决于节点数量以及他们之间如何互联
- Intel QuickPath Interconnect (QPI)

r3.8xlarge



x1.32xlarge


32 vCPU's
32 vCPU's
32 vCPU's
32 vCPU's
QPI
QPI
QPI
QPI
QPI
488GB
488GB
488GB
488GB

性能提示: NUMA平衡

- 相同NUMA节点上的进程，对本地内存的访问性能最优
- NUMA平衡是指内核把进程调度到离程序内存近的CPU上运行.
- Linux 3.8+版本支持自动NUMA平衡
- Windows Server 2003 企业版和数据中心版首次支持NUMA

- 如果你的应用使用的内存总量超过NUMA单节点的内存，或者应用线程在多个NUMA节点之间来回切换执行：
  - 设置内核启动参数“numa=off”来减少无效的内存拷贝
  - 使用 numactl 来绑定进程到指定NUMA节点的CPU

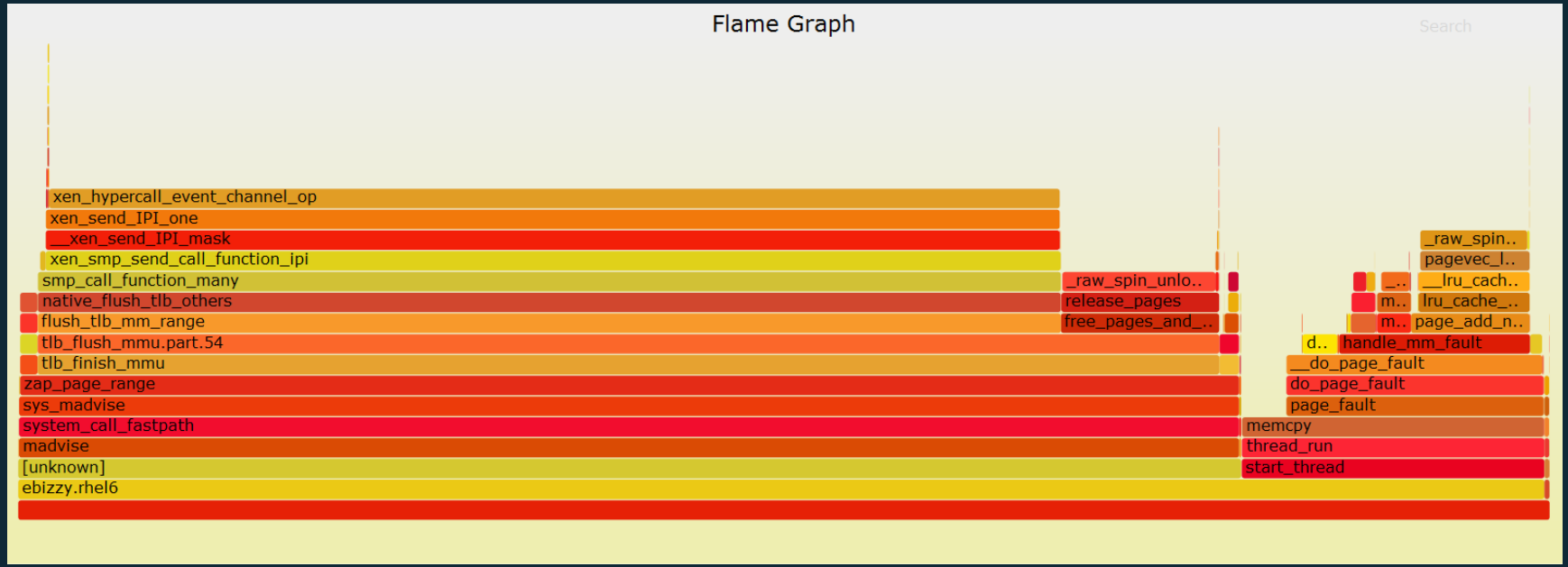
操作系统版本对性能的影响

- 例如，内存密集型的web应用
  - 创建很多线程
  - 快速的分配 / 回收内存
- 对比在CentOS 6 和 CentOS 7上的性能
- top命令显示了很大一部分CPU周期都消耗在内核态
- 测试工具ebizzy和客户的应用的行为类似
- perf命令来追踪和分析程序的性能

CentOS 6

```
[ec2-user@ip-172-31-12-150-CentOS6 ebizzy-0.3]$ sudo perf stat ./ebizzy -S 10
12,409 records/s
real 10.00 s
user  7.37 s
sys  341.22 s

Performance counter stats for './ebizzy -S 10':

     361458.371052      task-clock (msec)         #   35.880 CPUs utilized
           10,343      context-switches          #    0.029 K/sec
            2,582      cpu-migrations            #    0.007 K/sec
        1,418,204      page-faults               #    0.004 M/sec

      10.074085097 seconds time elapsed
```

火焰图 - CentOS 6



www.brendangregg.com/flamegraphs.html

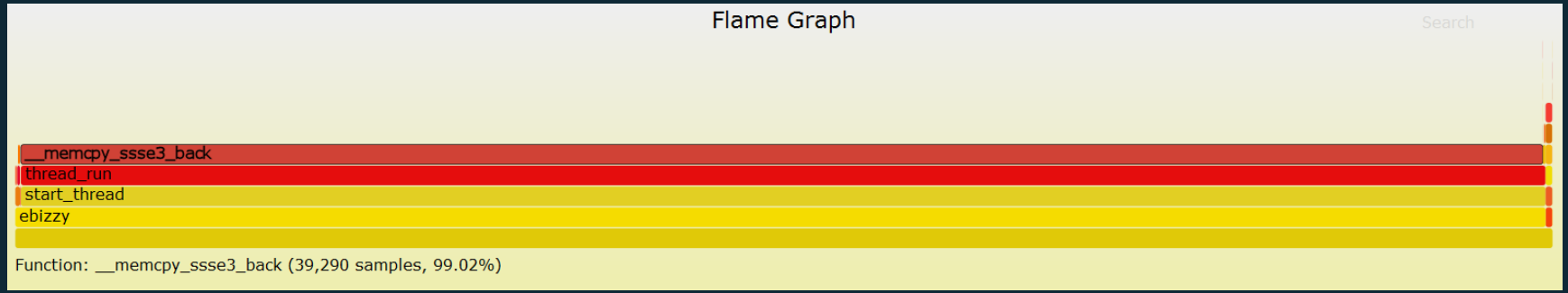
CentOS 7

```
[ec2-user@ip-172-31-7-22-CentOS7 ~]$ sudo perf stat ./ebizzy-0.3/ebizzy -S 10
425,143 records/s
real 10.00 s
user 397.28 s
sys   0.18 s

Performance counter stats for './ebizzy-0.3/ebizzy -S 10':

     397515.862535      task-clock (msec)         #   39.681 CPUs utilized
            25,256      context-switches          #    0.064 K/sec
             2,201      cpu-migrations            #    0.006 K/sec
            14,109      page-faults               #    0.035 K/sec

      10.017856000 seconds time elapsed
```

Up from 12,400 records/s!

Down from 1,418,204!

火焰图 - CentOS 7


Flame Graph
Search
__memcpy_ssse3_back
thread_run
start_thread
ebizzy
Function: __memcpy_ssse3_back (39,290 samples, 99.02%)

大页 Hugepages

- 禁用透明大页 Transparent Hugepages

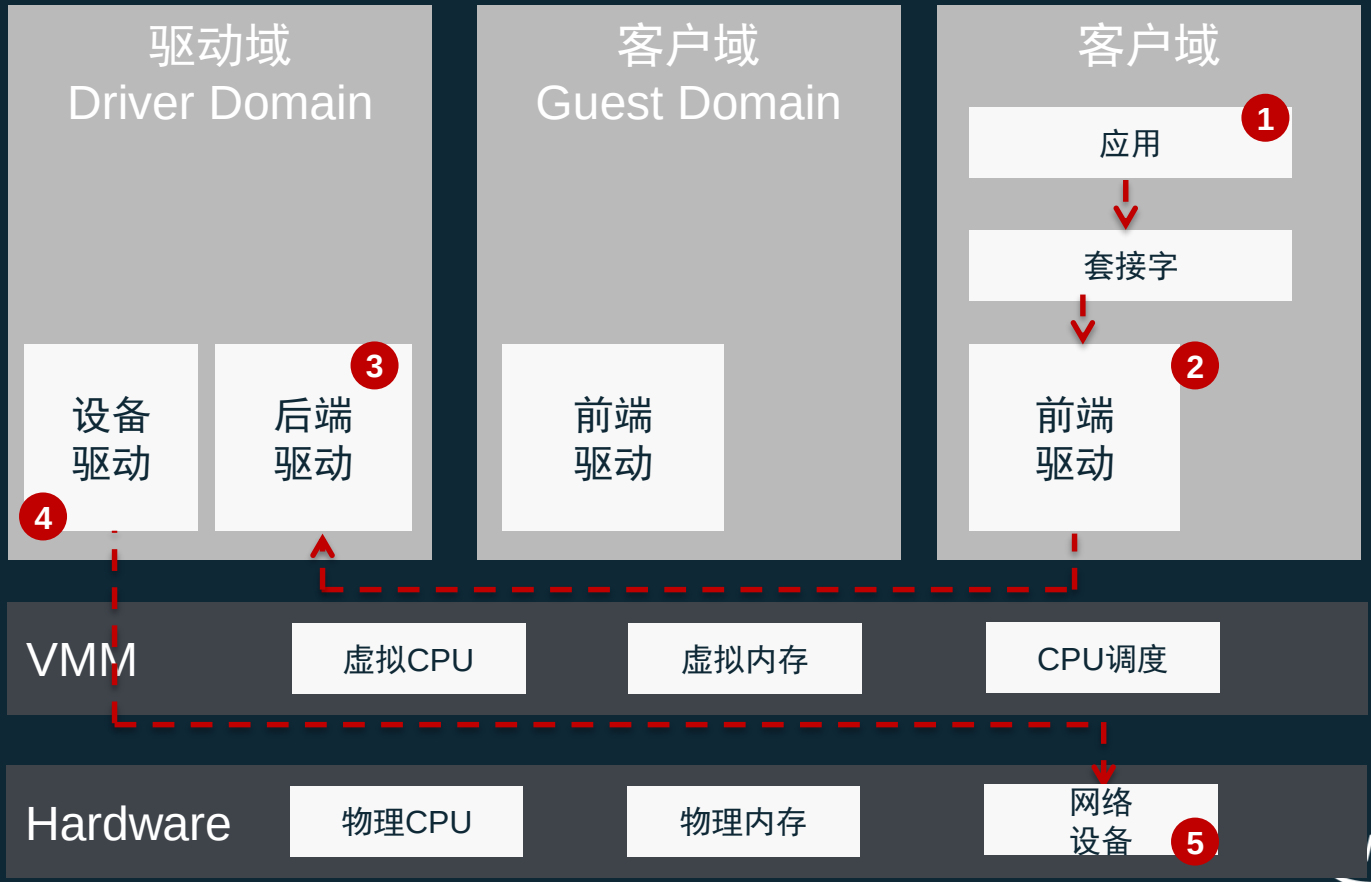
```
# echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled
# echo never > /sys/kernel/mm/redhat_transparent_hugepage/defrag
```

- 使用显式大页 Explicit Huge Pages

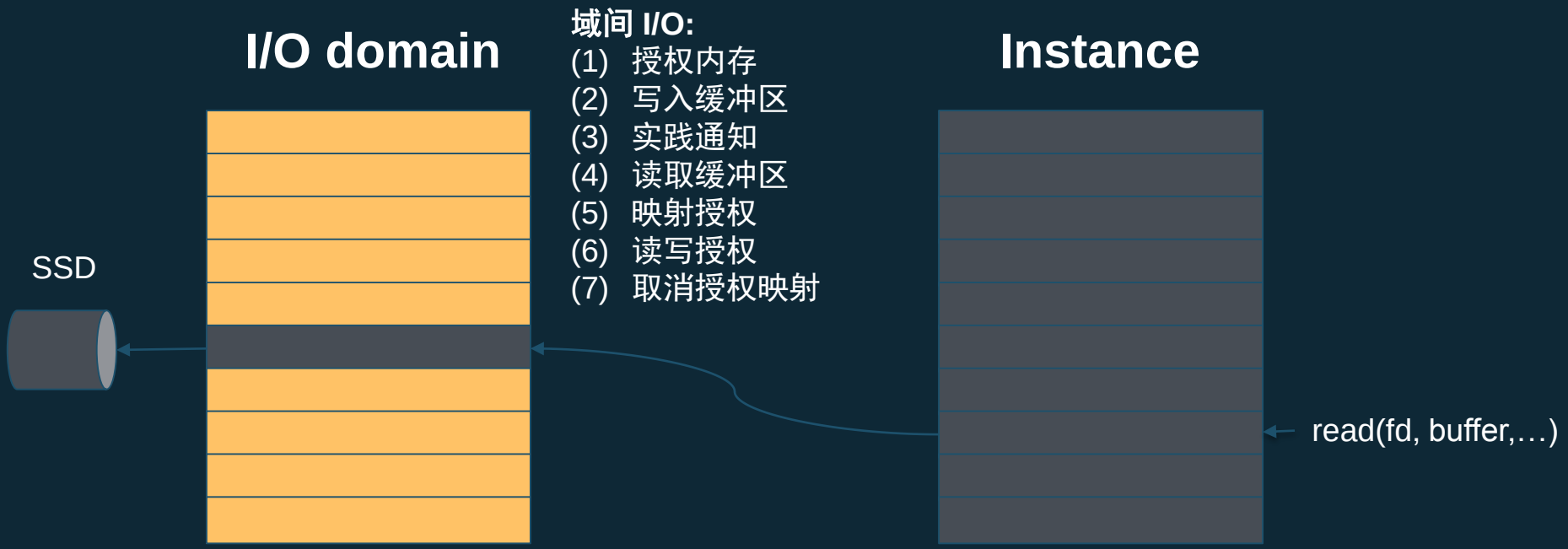
```
$ sudo mkdir /dev/hugetlbfs
$ sudo mount -t hugetlbfs none /dev/hugetlbfs
$ sudo sysctl -w vm.nr_hugepages=10000
$ HUGETLB_MORECORE=yes LD_PRELOAD=libhugetlbfs.so numactl --cpunodebind=0 \
     --membind=0 /path/to/application
```

参考: https://lwn.net/Articles/375096/

Xen分离驱动模型

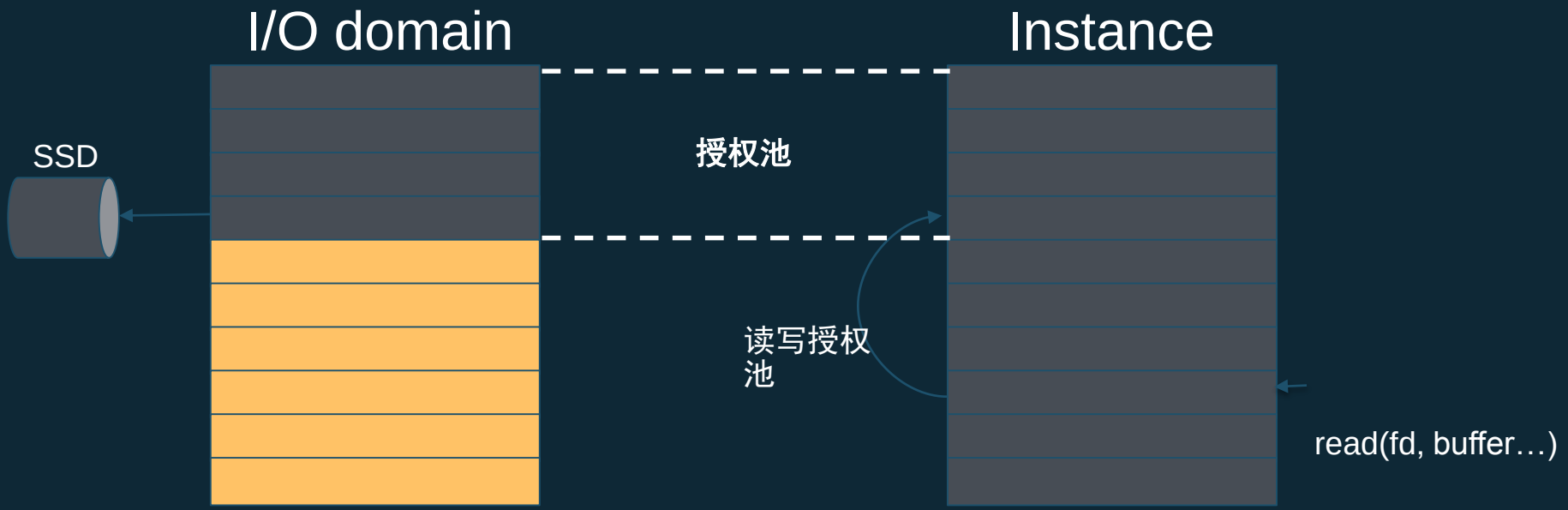

驱动域
Driver Domain
客户域
Guest Domain
客户域
1
应用
套接字
2
前端
驱动
3
后端
驱动
设备
驱动
4
前端
驱动
VMM
虚拟CPU
虚拟内存
CPU调度
Hardware
物理CPU
物理内存
网络
设备
5

Linux 3.8.0以前的版本



- 3.8.0以前的内核版本中需要授权映射“grant mapping”
- 授权映射的开销很大

Linux 3.8.0及以后的版本



- 批量建立持久的授权映射
- 从授权池里读取或者写入数据

验证是否启用了 Persistent Grants

[ec2-user@ip-172-31-4-129 ~]$ dmesg | egrep -i 'blkfront'

Blkfront and the Xen platform PCI driver have been compiled for this kernel: unplug emulated disks.

blkfront: xvda: barrier or flush: disabled; persistent grants: enabled; indirect descriptors: enabled;

blkfront: xvdb: flush diskcache: enabled; ; indirect descriptors: enabled;

blkfront: xvdc: flush diskcache: enabled; persistent grants: enabled; indirect descriptors: enabled;

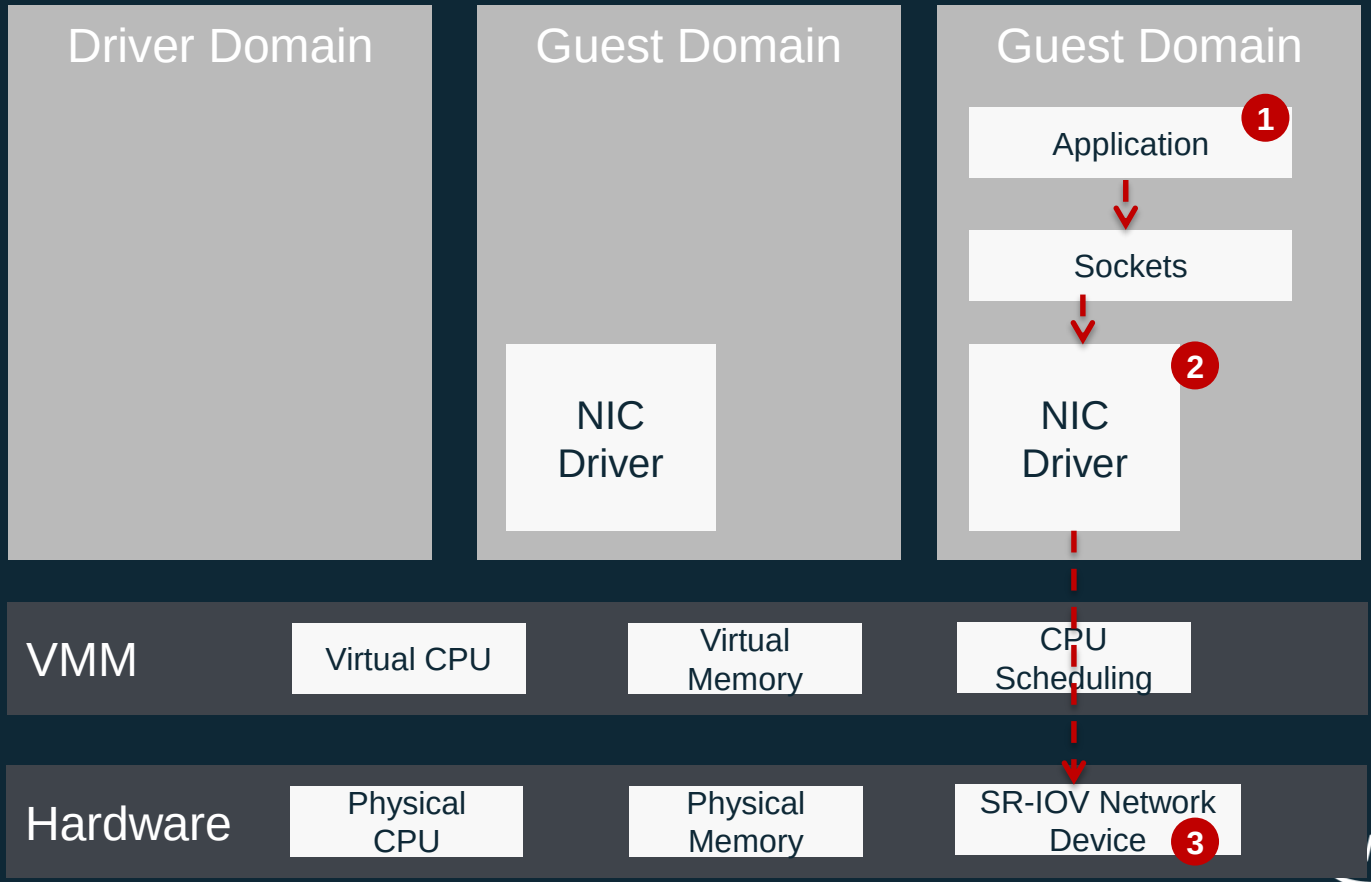
性能提示: 使用3.10+版本的Linux内核

- Amazon Linux 13.09 or later
- Ubuntu 14.04 or later
- RHEL/Centos 7 or later
- 其他

设备直通: 增强型联网Enhanced Networking

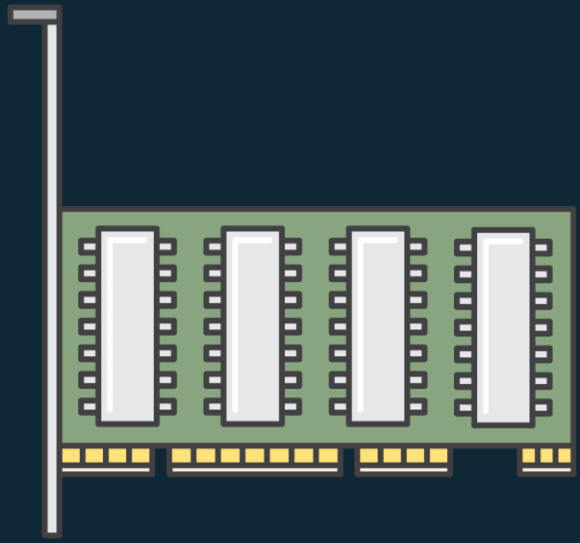
- SR-IOV 不再需要Hypervisor的驱动中转
- 物理网络设备将虚拟化功能暴露给EC2实例
- 需要特殊的驱动程序:
  - 实例的操作系统需要支持ixgbevf / ena驱动
  - 标记AMI对增强型联网的支持

增强型联网


Driver Domain
Guest Domain
Guest Domain
Application
1
Sockets
NIC Driver
2
NIC Driver
VMM
Virtual CPU
Virtual Memory
CPU Scheduling
Hardware
Physical CPU
Physical Memory
SR-IOV Network Device
3

弹性网络适配器 Elastic Network Adapter

- 下一代增强型联网 Enhanced Networking
  - 硬件校验和
  - 多发送、接收队列
  - Receive Side Steering
- 25Gbps
- 全新的开源驱动



网络带宽

- 25 Gbps 和 10 Gbps 的实例间最大带宽
  - 单向带宽，双向带宽=单项带宽x2
  - 置放组(placement groups) 内，稳定的低延时
- 高, 中等, 低–使用iperf3 进行带宽测试
- 目前，单个实例到置放组外的实例的带宽最高5 Gbps
  - 包括到S3或者Internet带宽

EBS 吞吐和IOPS性能

- 实例规格的EBS吞吐上限
- EBS卷的容量和类型
- 启用EBS优化特性

Instance type	Amazon EBS-optimized by default	Max. bandwidth (Mbps)*	Expected throughput (MB/s)**	Max. IOPS (16 KB I/O size)**
c4.8xlarge	Yes	4,000	500	32,000
d2.8xlarge	Yes	4,000	500	32,000
f1.16xlarge	Yes	14,000	1,750	75,000
g3.16xlarge	Yes	14,000	1,750	80,000
i3.16xlarge	Yes	14,000	1,750	65,000
m4.10xlarge	Yes	4,000	500	32,000
m4.16xlarge	Yes	10,000	1,250	65,000
p2.16xlarge	Yes	10,000	1,250	65,000
p3.16xlarge	Yes	14,000	1,750	80,000
r4.16xlarge	Yes	14,000	1,750	75,000
x1.32xlarge	Yes	14,000	1,750	80,000
x1e.32xlarge	Yes	14,000	1,750	80,000

完整列表请参考
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSOptimized.html

Summary: Getting the Most Out of EC2 Instances

- 选择最新世代的实例，使用HVM AMI
- 配置时钟源为TSC
- 配置处理器核心的P状态和C状态
- 监控T2实例的CPU额度
- 使用新版本的操作系统和内核
- NUMA配平
- 确保Persistent grants生效
- 使用支持增强型联网的实例和AMI
- 对实际应用进行性能分析Profiling

虚拟化的目标

- 与裸机相当的性能指标, 很多场景下我们已经做到了
- 一部 Hypervisor 和 Driver Domain的消亡史
  - 硬件辅助虚拟化HVM
  - 调度和 granting 效率
  - 设备直通

下一步

- 参考 Amazon EC2 文档
- 创建实例，运行您的应用!

谢谢！