Deep Dive on Amazon EC2 Instances

Featuring Performance Optimization Best Practices

Julien Simon
Principal Technical Evangelist, AWS
julsimon@amazon.fr
@julsimon

What to Expect from the Session

- 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 Elastic Compute Cloud Is Big

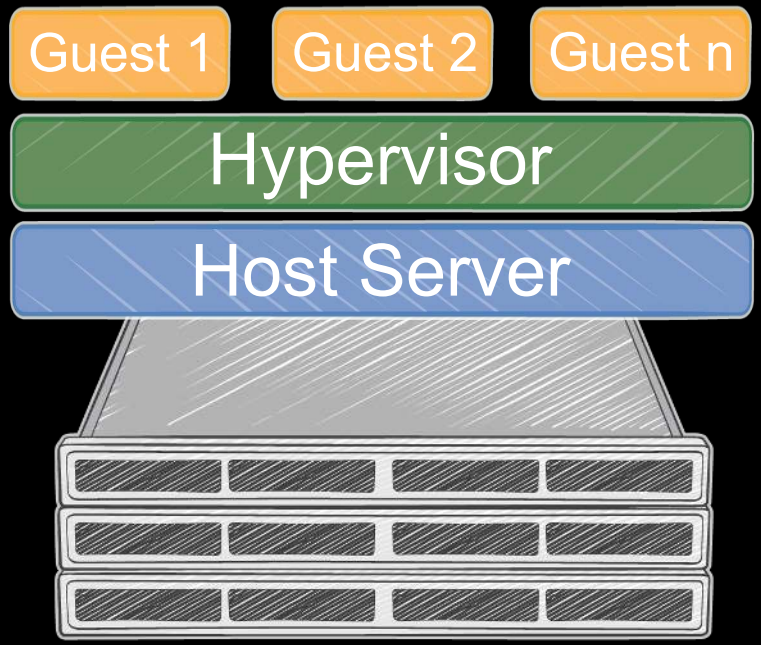
Purchase options

API

Instances

Networking

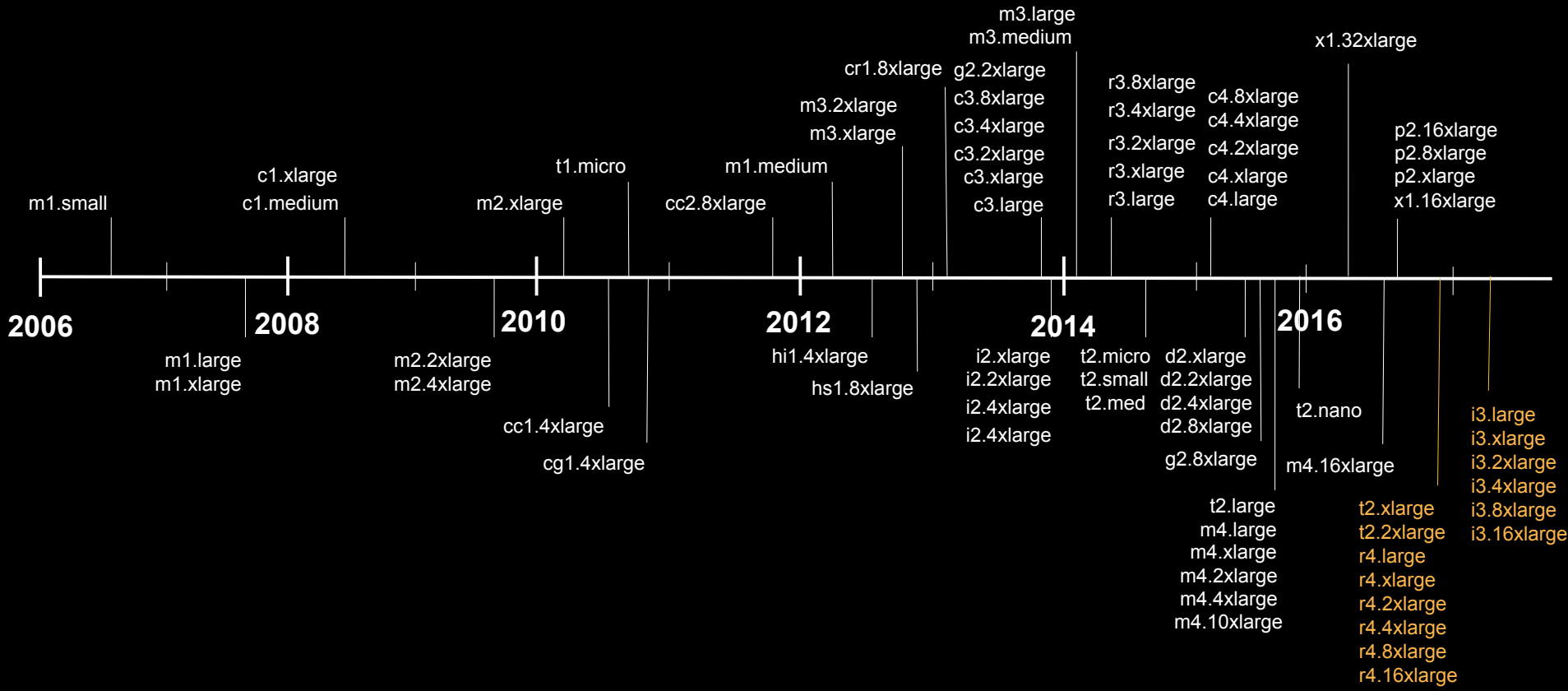
Amazon EC2 Instances



In the past

- First launched in August 2006
- M1 instance
  - “One size fits all”

Amazon EC2 Instances History

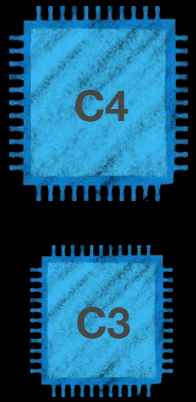



Instance generation
c4.xlarge
Instance family
Instance size

EC2 Instance Families

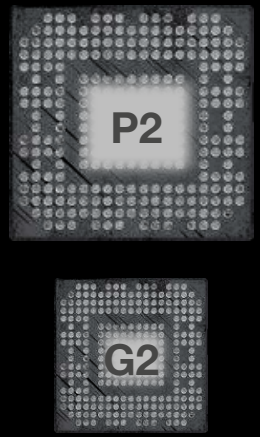
General purpose

Compute optimized

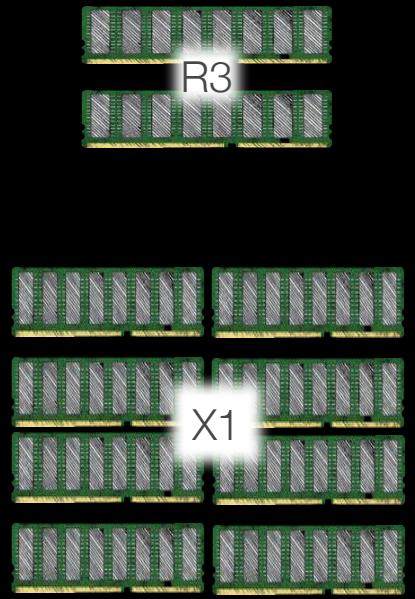


Storage and I/O optimized

GPU optimized



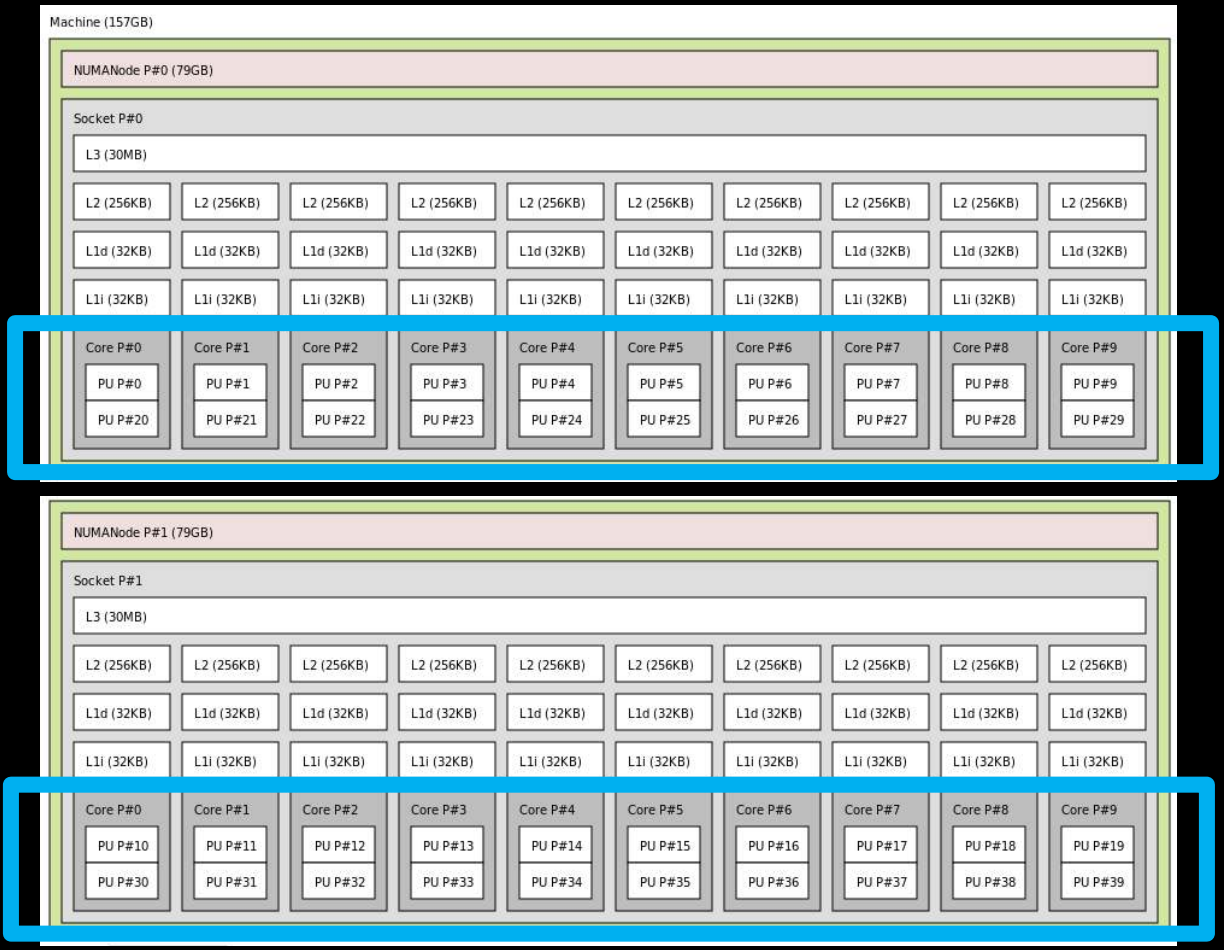
Memory optimized



What's a Virtual CPU? (vCPU)

- A vCPU is typically a hyper-threaded physical core*
  - On Linux, "A" thread ids enumerated before "B" thread ids
    - A: 0-3, B: 4-7
  - On Windows, thread ids are interleaved
    - A: 0, 2, 4, 6. B : 1, 3, 5, 7
- Divide vCPU count by 2 to get core count
- Cores by EC2 & RDS DB Instance type: https://aws.amazon.com/ec2/virtualcores/
- "Demystifying the Number of vCPUs for Optimal Workload Performance" https://d0.awsstatic.com/whitepapers/Demystifying_vCPUs.pdf

* Except on the "t" family and m3.medium



‘lstopo’ output for m4.10xlarge: 40 threads and 20 physical cores

Disable Hyper-Threading If You Need To

- Useful for FPU heavy applications

Linux

- Use ‘lscpu’ to validate layout
- Hot offline the “B” threads

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

- Set grub to only initialize the first half of all threads

```
maxcpus=63
```

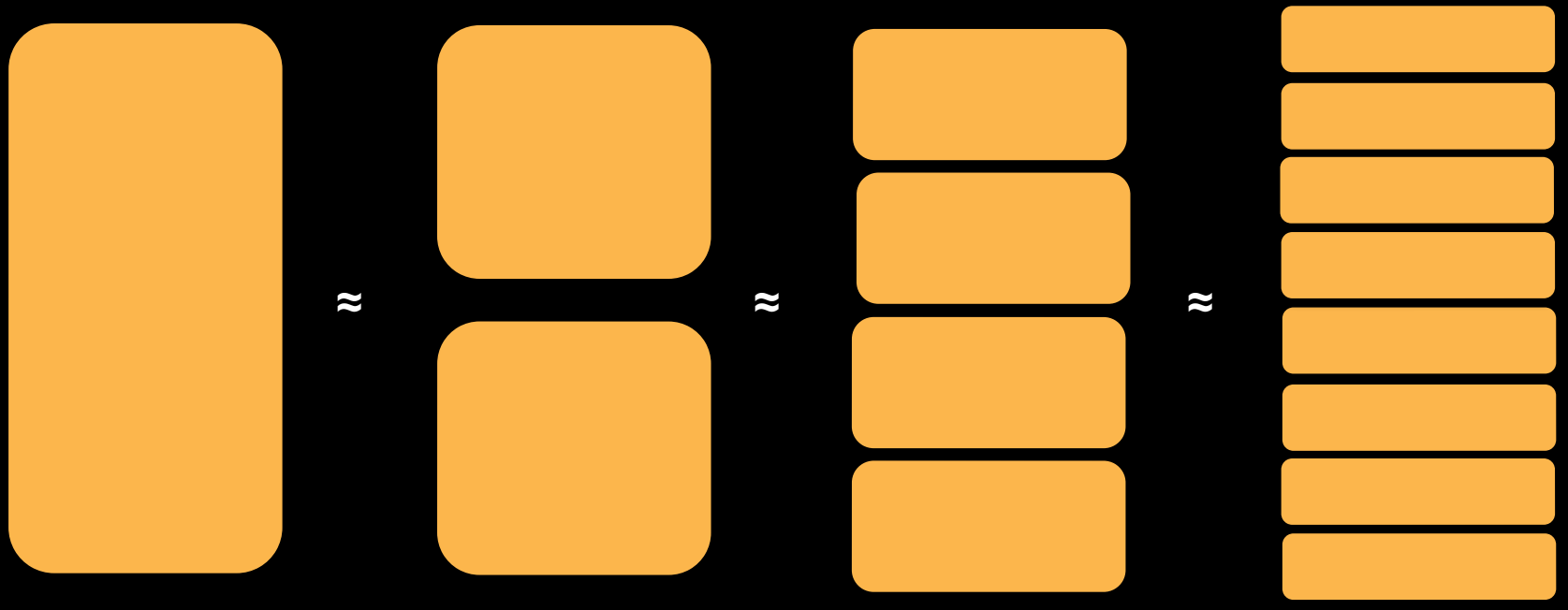
```
[ec2-user@ip-172-31-7-218 ~]$ lscpu
CPU(s):                  128
On-line CPU(s) list:     0-127
Thread(s) per core:      2
Core(s) per socket:      16
Socket(s):               4
NUMA node(s):            4
Model name:              Intel(R) Xeon(R) CPU
Hypervisor vendor:       Xen
Virtualization type:     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
```

Windows

- More complicated due to interleaved thread ids
- Use “CPU affinity”

‘lstopo’ output for m4.10xlarge with HT disabled: 20 threads and 20 physical cores

Instance sizing



c4.8xlarge

2 - c4.4xlarge

4 - c4.2xlarge

8 - c4.xlarge

Resource Allocation

- All resources assigned to you are dedicated to your instance with no over commitment*
  - All vCPUs are dedicated to you
  - Memory allocated is assigned only to your instance
  - Network resources are partitioned to avoid “noisy neighbors”
- Curious about the number of instances per host? Use “Dedicated Hosts” as a guide.

*Again, the “t” family is special

“Launching new instances and running tests in parallel is easy…[when choosing an instance] there is no substitute for measuring the performance of your full application.”

- EC2 documentation

Timekeeping Explained

- Timekeeping in an instance is deceptively hard
  - gettimeofday(), clock_gettime(), QueryPerformanceCounter()
- Xen clock
  - Handled by the Xen hypervisor
  - Does not support vDSO (virtual dynamic shared object)

  → Requires a system call, leading to context switches, etc. → Slow
- Time Stamp Counter (TSC)
  - CPU counter, accessible from userspace through vDSO → Much faster
  - Available on Sandy Bridge processors and newer
- On current generation instances, use TSC as clocksource

Benchmarking - Time Intensive Application

```
#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;
}
```

Using the Xen Clock Source

```
[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
```

Using the TSC Clock Source

```
[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
```

Tip: Use TSC as clocksource

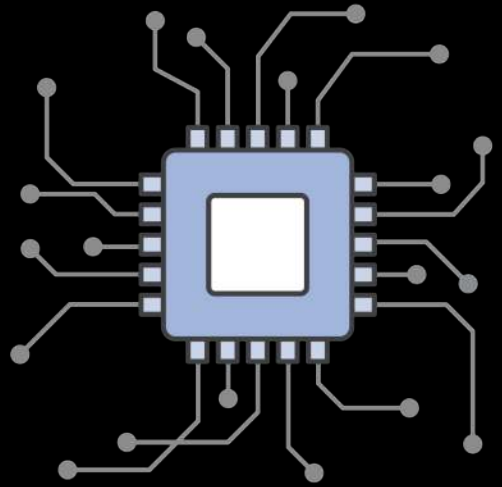
```
# cat /sys/devices/system/cl*/cl*/available_clocksource
xen tsc hpet acpi_pm

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

Change with:

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

P-state and C-state Control

- c4.8xlarge, d2.8xlarge, m4.10xlarge, m4.16xlarge, p2.16xlarge, x1.16xlarge, x1.32xlarge
- By entering deeper idle states, non-idle cores can achieve up to 300MHz higher clock frequencies
- But… deeper idle states require more time to exit, may not be appropriate for latency-sensitive workloads
- Limit c-state by adding “intel_idle.max_cstate=1” to grub



https://aws.amazon.com/blogs/aws/now-available-new-c4-instances/
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/processor_state_control.html

Tip: P-state Control for AVX2

- If an application makes heavy use of AVX2 on all cores, the processor may attempt to draw more power than it should
- Processor will transparently reduce frequency
- Frequent changes of CPU frequency can slow an application

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

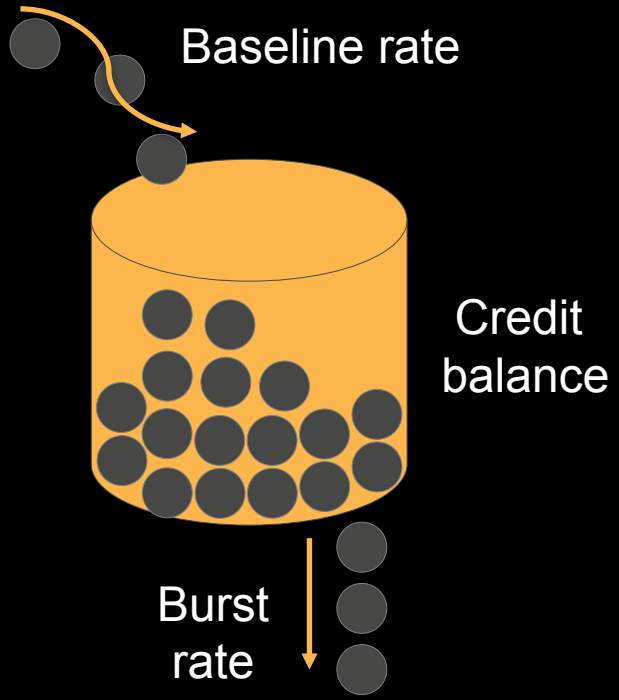
Review: T2 Instances

- Lowest cost EC2 instance at $0.0059 per hour
- Burstable performance
- Fixed allocation enforced with CPU credits

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

General purpose, web serving, developer environments, small databases

How Credits Work



- A CPU credit provides the performance of a full CPU core for one minute
- An instance earns CPU credits at a steady rate
- An instance consumes credits when active
- Credits expire (leak) after 24 hours

Tip: Monitor CPU Credit Balance

Review: X1 Instances

- Largest memory instance with 2 TB of DRAM
- Quad socket, Intel E7 processors with 128 vCPUs

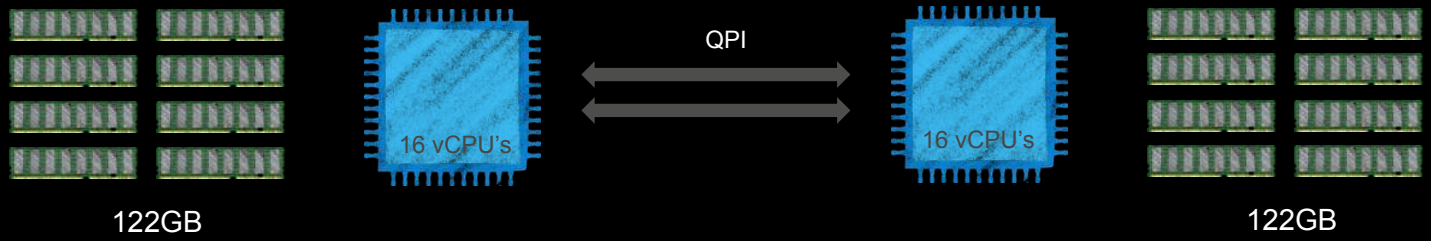
Model	vCPU	Memory (GiB)	Local Storage	Network
x1.16xlarge	64	976	1x 1920GB SSD	10Gbps
x1.32xlarge	128	1952	2x 1920GB SSD	20Gbps

In-memory databases, big data processing, HPC workloads

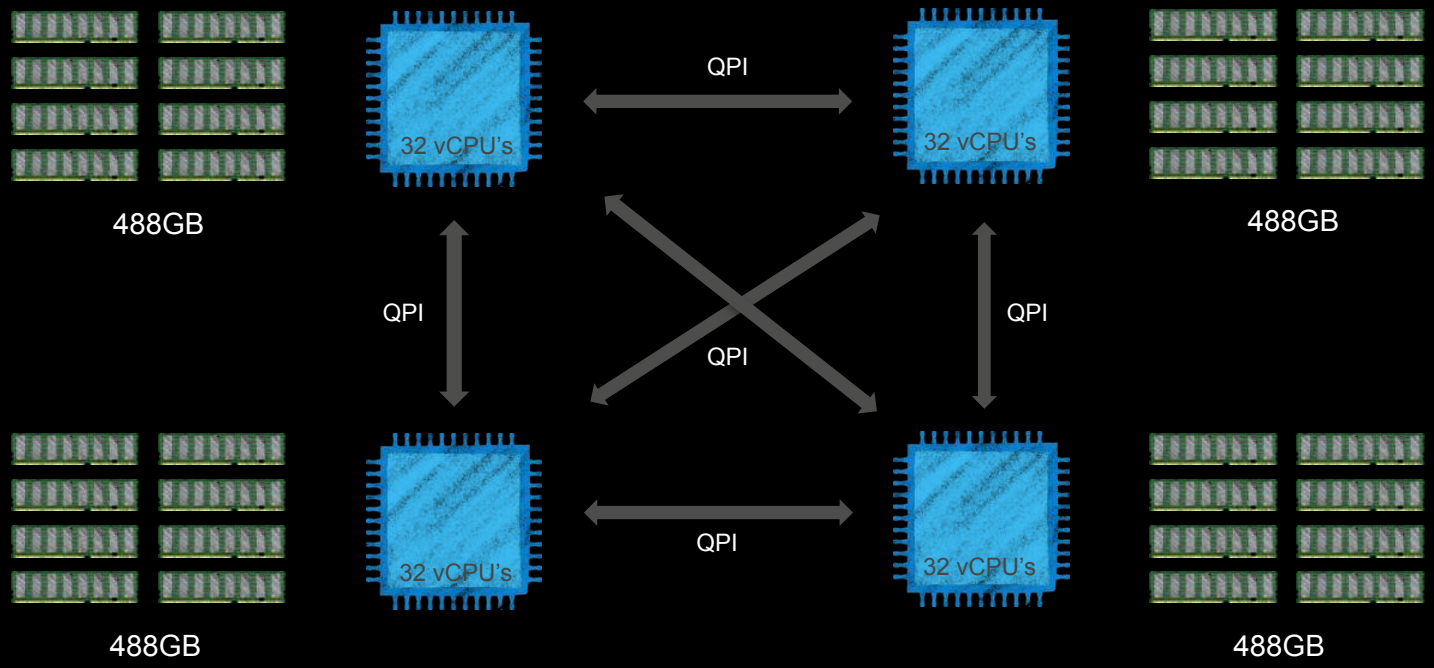
NUMA

- Non-uniform memory access
- Each processor in a multi-CPU system has local memory that is accessible through a fast interconnect
- Each processor can also access memory from other CPUs, but local memory access is a lot faster than remote memory
- Performance is related to the number of CPU sockets and how they are connected - Intel QuickPath Interconnect (QPI)

r3.8xlarge



x1.32xlarge


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

Tip: Kernel Support for NUMA Balancing

- An application will perform best when the threads of its processes are accessing memory on the same NUMA node.
- NUMA balancing moves tasks closer to the memory they are accessing.
- This is all done automatically by the Linux kernel when automatic NUMA balancing is active: version 3.8+ of the Linux kernel.
- Windows support for NUMA first appeared in the Enterprise and Data Center SKUs of Windows Server 2003.
- Set “numa=off” or use numactl to reduce NUMA paging if your application uses more memory than will fit on a single socket or has threads that move between sockets

Operating Systems Impact Performance

- Memory intensive web application
  - Created many threads
  - Rapidly allocated/deallocated memory
- Comparing performance of RHEL6 vs RHEL7
- Notice high amount of “system” time in top
- Found a benchmark tool (ebizzy) with a similar performance profile
- Traced it’s performance with “perf”

https://sourceforge.net/projects/ebizzy/
https://perf.wiki.kernel.org

On RHEL6

```
[ec2-user@ip-172-31-12-150-RHEL6 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
```

RHEL6 Flame Graph Output

www.brendangregg.com/flamegraphs.html

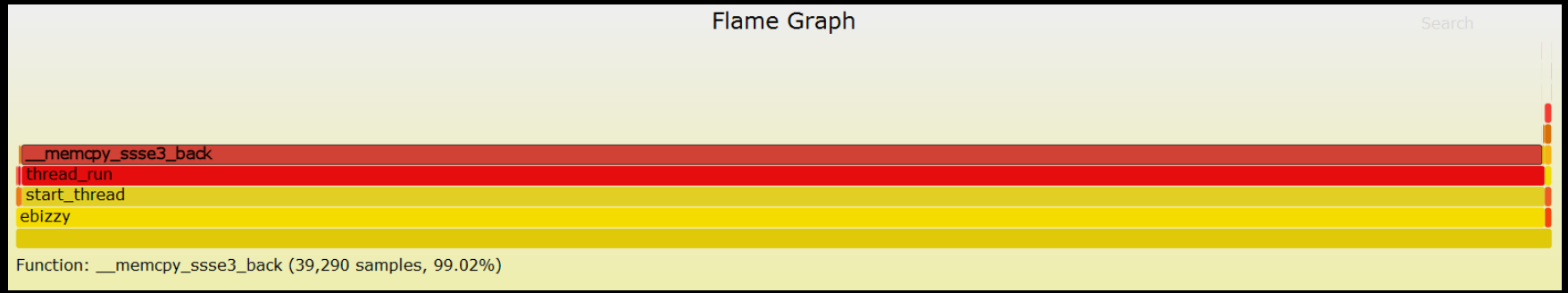
On RHEL7

```
[ec2-user@ip-172-31-7-22-RHEL7 ~]$ 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!

RHEL7 Flame Graph Output



Hugepages

- Disable Transparent Hugepages

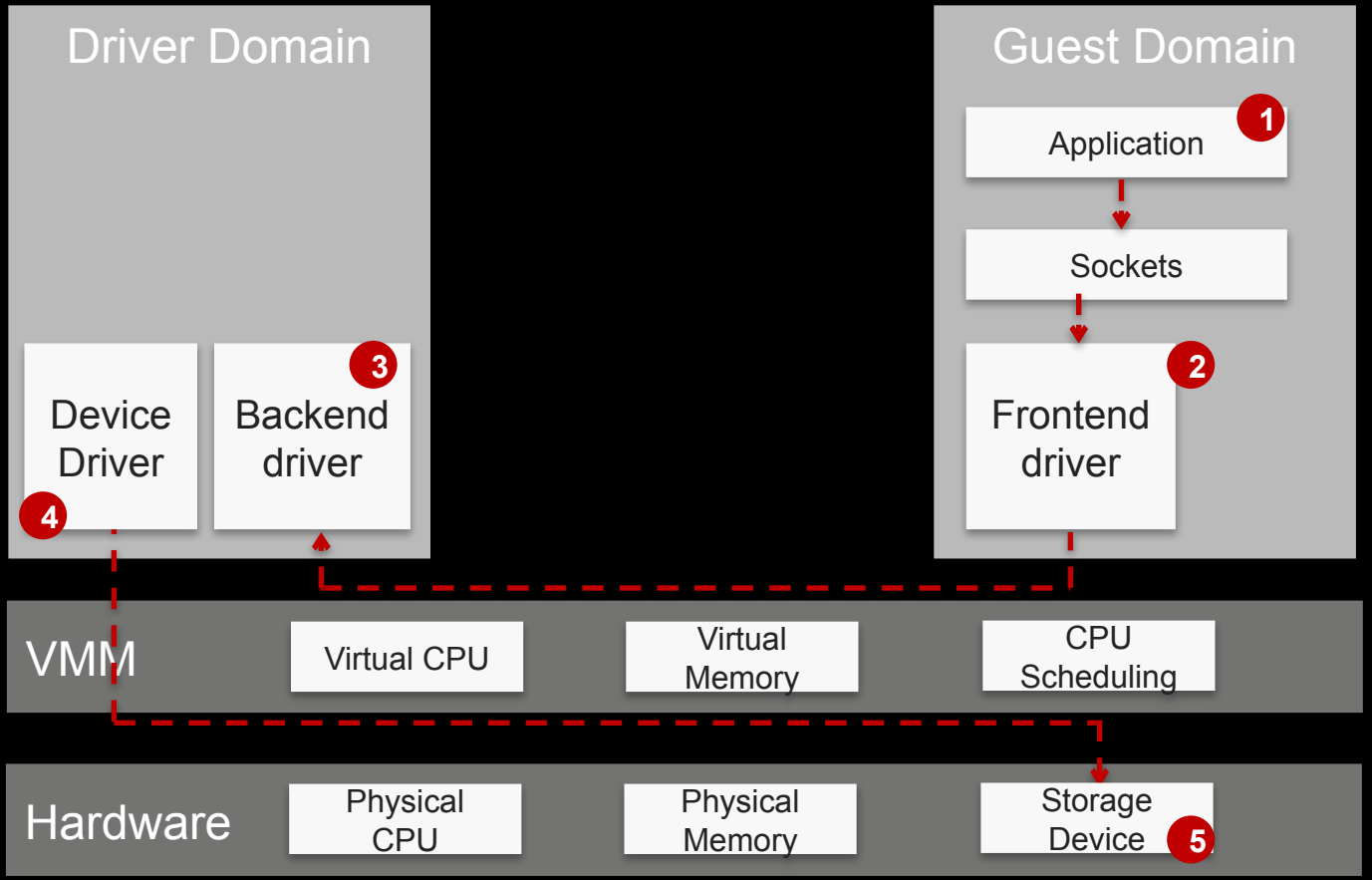
```
# echo never > /sys/kernel/mm/transparent_hugepage/enabled
# echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

- Use Explicit Huge Pages

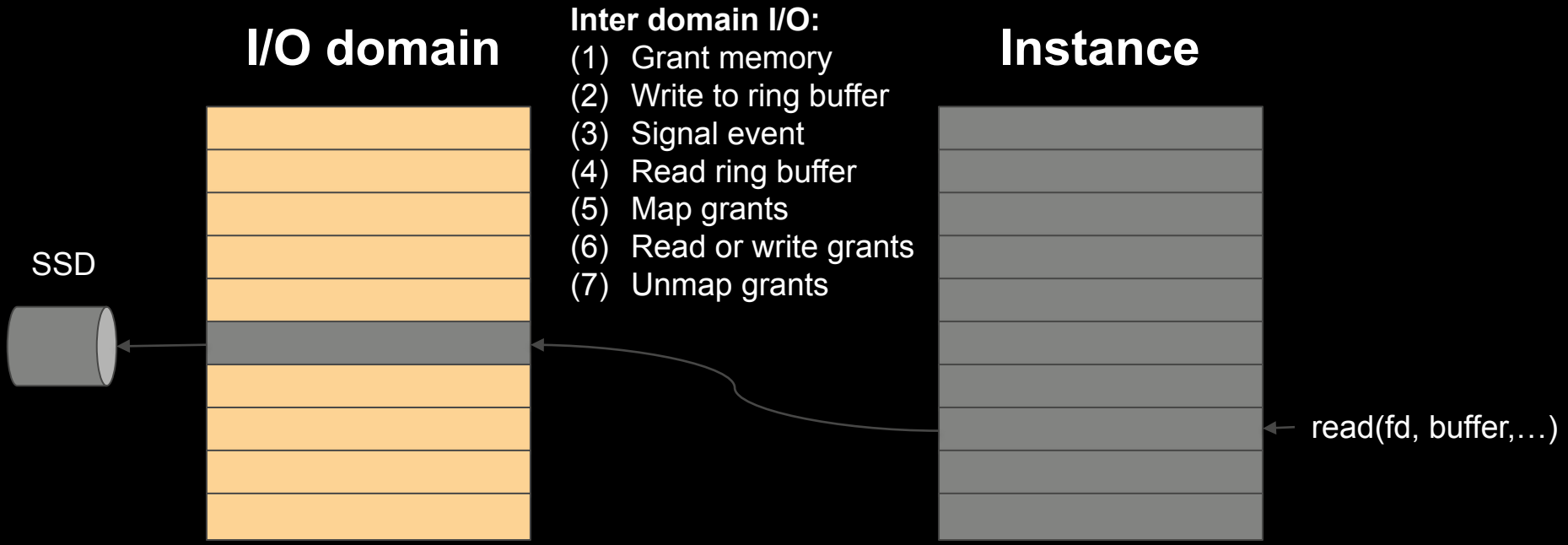
```
$ sudo mkdir /dev/hugetlbfs
$ sudo mount -t hugetlbfs none /dev/hugetlbfs
$ sudo sysctl -w vm.nr_hugepages=10000
$ HUGETLB_MORECORE=yes LD_PRELOAD=libhugetlbfs.so numactl --cpunodebind=0 \
     --membind=0 /path/to/application
```

https://lwn.net/Articles/375096/

Split Driver Model



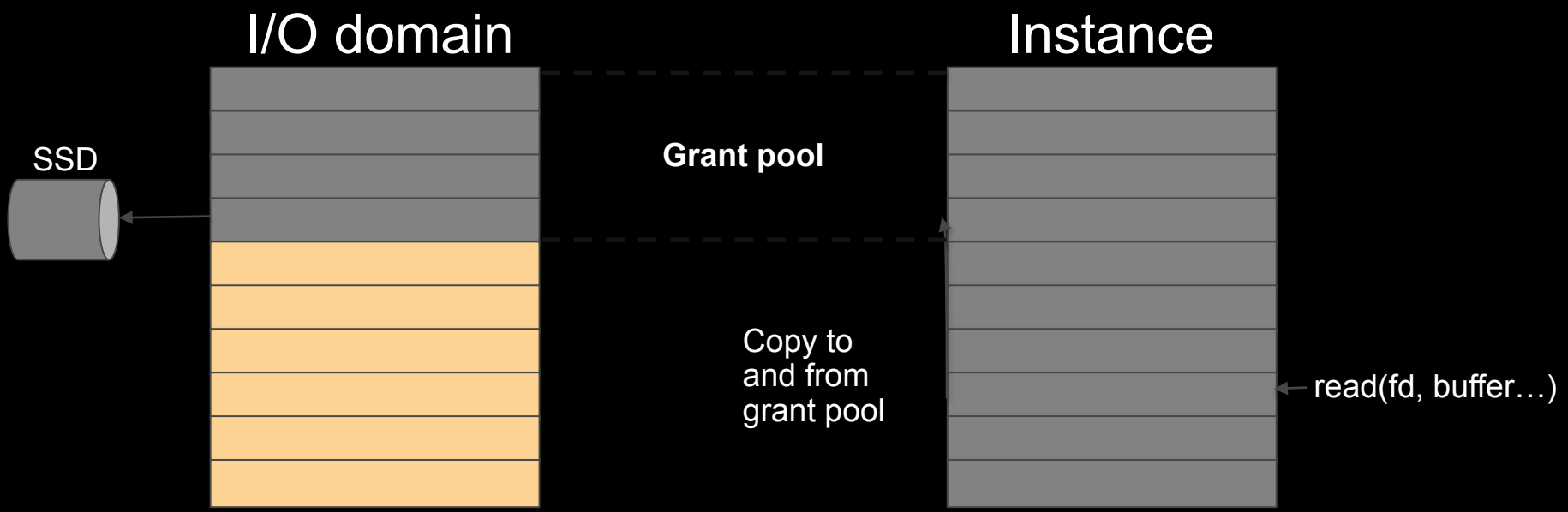
Granting in pre-3.8.0 Kernels



- Requires “grant mapping” prior to 3.8.0
- Grant mappings are expensive operations due to TLB flushes

https://blog.xenproject.org/2012/11/23/improving-block-protocol-scalability-with-persistent-grants/

Persistent granting in 3.8.0+ Kernels



- Grant mappings are set up in a pool one time
- Data is copied in and out of the grant pool

Validating 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; persistent grants: enabled; indirect descriptors: enabled;
blkfront: xvdc: flush diskcache: enabled; persistent grants: enabled; indirect descriptors: enabled;
blkfront: xvdd: flush diskcache: enabled; persistent grants: enabled; indirect descriptors: enabled;
blkfront: xvde: flush diskcache: enabled; persistent grants: enabled; indirect descriptors: enabled;
blkfront: xvdf: flush diskcache: enabled; persistent grants: enabled; indirect descriptors: enabled;
blkfront: xvdg: flush diskcache: enabled; persistent grants: enabled; indirect descriptors: enabled;
blkfront: xvdh: flush diskcache: enabled; persistent grants: enabled; indirect descriptors: enabled;
blkfront: xvdi: flush diskcache: enabled; persistent grants: enabled; indirect descriptors: enabled;
```

2009 – Longer ago than you think

- Avatar was the top movie in the theaters
- Facebook overtook MySpace in active users
- President Obama was sworn into office
- **The 2.6.32 Linux kernel was released**

CentOS 6
Community ENTerprise Operating System

Tip: Use 3.10+ kernel

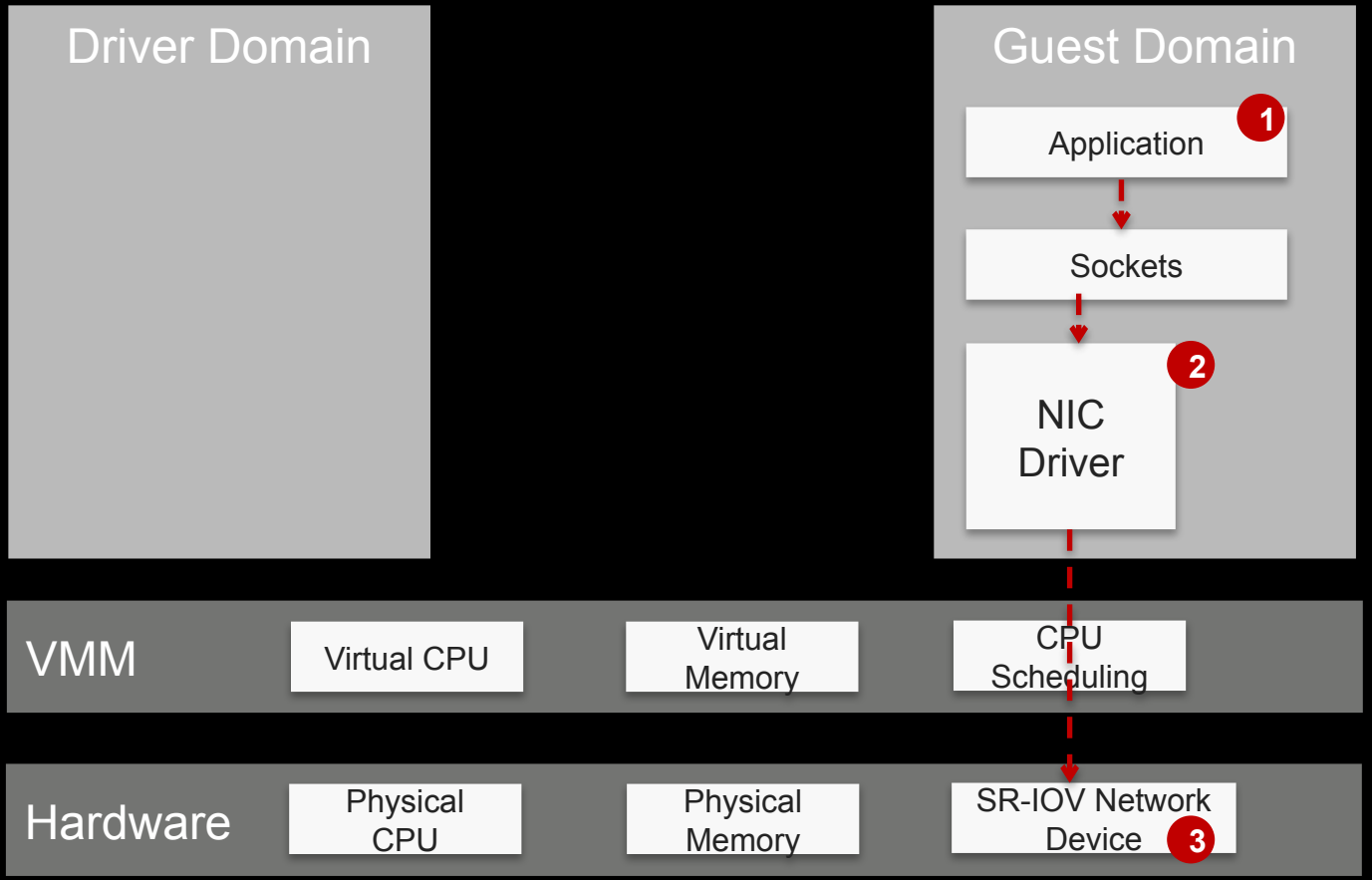
- Amazon Linux 2013.09 or later
- Ubuntu 14.04 or later
- RHEL/Centos 7 or later
- Etc.

Device Pass Through: Enhanced Networking

- SR-IOV eliminates need for driver domain
- Physical network device exposes virtual function to instance
- Requires a specialized driver, which means:
  - Your instance OS needs to know about it
  - EC2 needs to be told your instance can use it

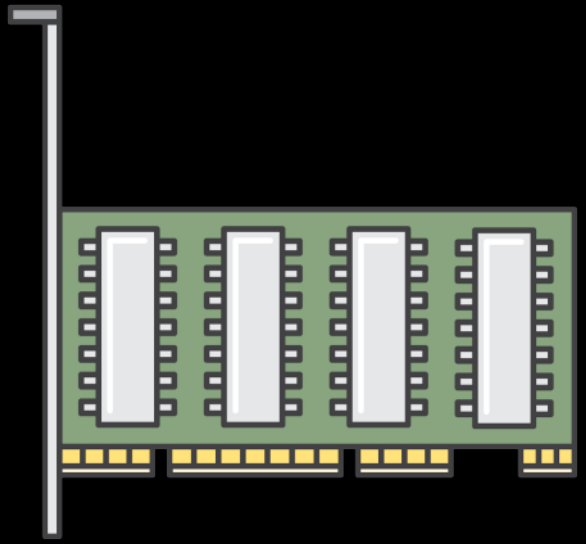
More information in our previous “Deep Dive VPC” webinar
https://www.youtube.com/watch?v=hUw4ehDswWo

After Enhanced Networking



Elastic Network Adapter

- Next Generation of Enhanced Networking
  - Hardware Checksums
  - Multi-Queue Support
  - Receive Side Steering
- 20Gbps in a Placement Group
- New Open Source Amazon Network Driver



https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking-ena.html
https://github.com/amzn/amzn-drivers

Network Performance

- Use placement groups when you need high and consistent instance to instance bandwidth
- 20 Gigabit & 10 Gigabit
  - Measured one-way, double that for bi-directional (full duplex)
- High, Moderate, Low – A function of the instance size and EBS optimization
  - Not all created equal – Test with iperf if it's important!
- All traffic limited to 5 Gb/s when exiting EC2

EBS Performance

- Instance size affects throughput
- Match your volume size and type to your instance
- Use EBS optimization if EBS performance is important

Instance type	EBS-optimized by default	Max. bandwidth (MiB/s)*	Max. IOPS (16 KiB I/O size)*	Throughput (Mbps)**
c3.xlarge		62.5	4,000	500
c3.2xlarge		125	8,000	1,000
c3.4xlarge		250	16,000	2,000
c4.large	Yes	62.5	4,000	500
c4.xlarge	Yes	93.75	6,000	750
c4.4xlarge	Yes	250	16,000	2,000
c4.8xlarge	Yes	500	32,000	4,000
d2.xlarge	Yes	93.75	6,000	750
d2.2xlarge	Yes	125	8,000	1,000
d2.4xlarge	Yes	250	16,000	2,000
d2.8xlarge	Yes	500	32,000	4,000
g2.2xlarge		125	8,000	1,000
i2.xlarge		62.5	4,000	500
i2.2xlarge		125	8,000	1,000
i2.4xlarge		250	16,000	2,000
m3.xlarge		62.5	4,000	500
m3.2xlarge		125	8,000	1,000
m4.large	Yes	56.25	3,600	450
m4.xlarge	Yes	93.75	6,000	750
m4.2xlarge	Yes	125	8,000	1,000
m4.4xlarge	Yes	250	16,000	2,000
m4.10xlarge	Yes	500	32,000	4,000

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-ec2-config.html

Summary: Getting the Most Out of EC2 Instances

- Choose HVM AMIs
- Timekeeping: use TSC
- C state and P state controls
- Monitor T2 CPU credits
- Use a modern Linux OS
- NUMA balancing
- Persistent grants for I/O performance
- Enhanced networking
- Profile your application!

AWS User Groups

Lille
Paris
Rennes
Nantes
Bordeaux
Lyon
Montpellier
Toulouse
Côte d'Azur (new!)

facebook.com/groups/AWSFrance/

@aws_actus

https://aws.amazon.com/fr/events/webinaires/

Avril
Mardi 25 avril - 15h30
Les bases de données relationnelles (en savoir plus)

Mai
Mardi 30 mai - 15h30
Le Big Data (en savoir plus)

Juin
Mardi 27 juin - 15h30
Les fondamentaux AWS (en savoir plus)

Chaîne “Amazon Web Services France” sur YouTube
https://www.youtube.com/channel/UCDE2Dt16Asi-RiR_GNe9scA

Thank you!

Julien Simon
Principal Technical Evangelist, AWS
julsimon@amazon.fr
@julsimon