APMCon

AWS 云计算之上 Linux 实例的优化

费良宏，lianghon@amazon.com

AWS 首席云计算技术顾问

August 19, 2016

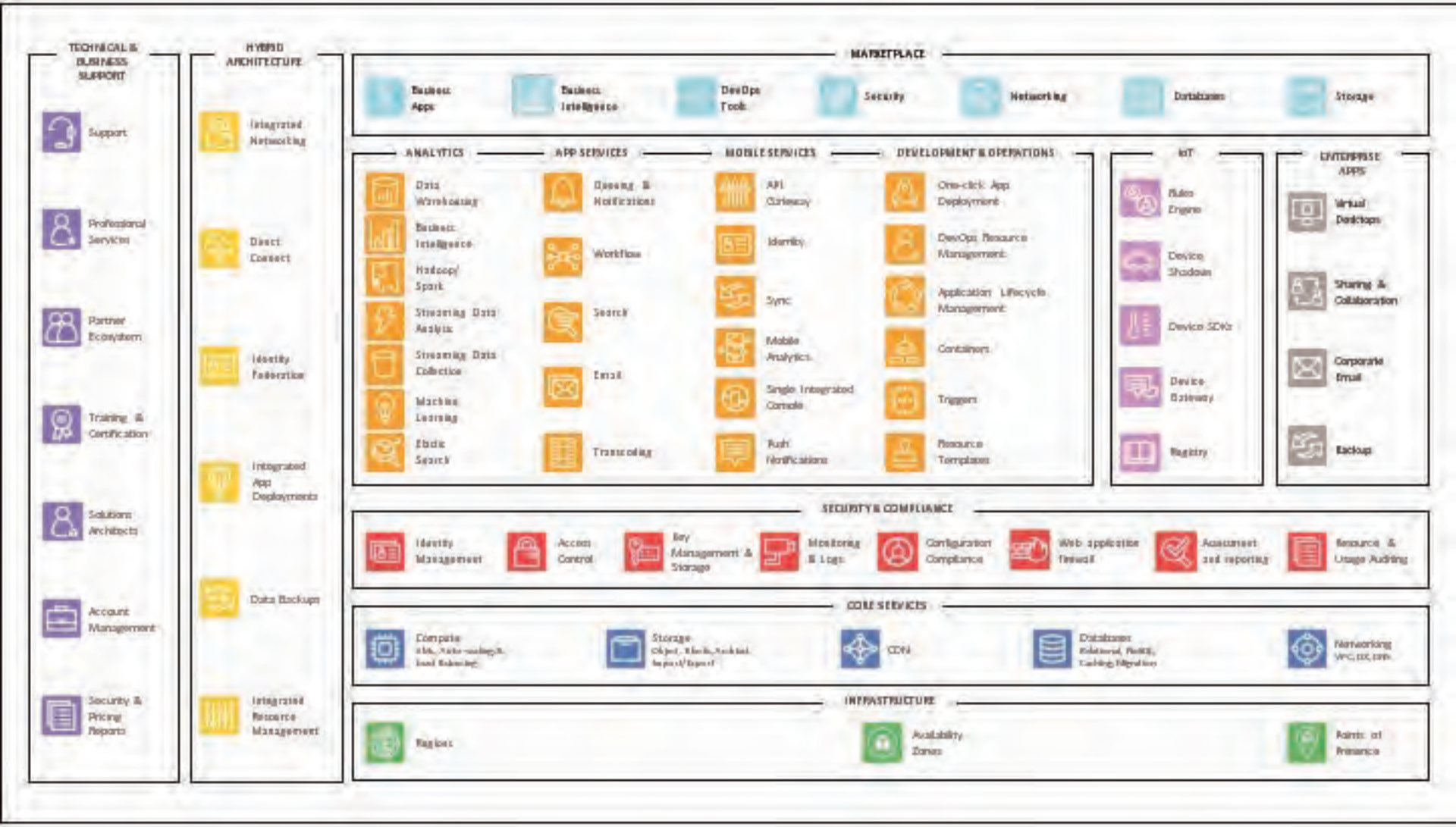
AWS 全球基础设施

APMCon

13 个区域、35 个可用区、56 个边缘站点

AWS 广泛的云计算服务

APMCon



2016 Gartner 云计算 IaaS 魔力象限

APMCon

Figure 1. Magic Quadrant for Cloud Infrastructure as a Service, Worldwide

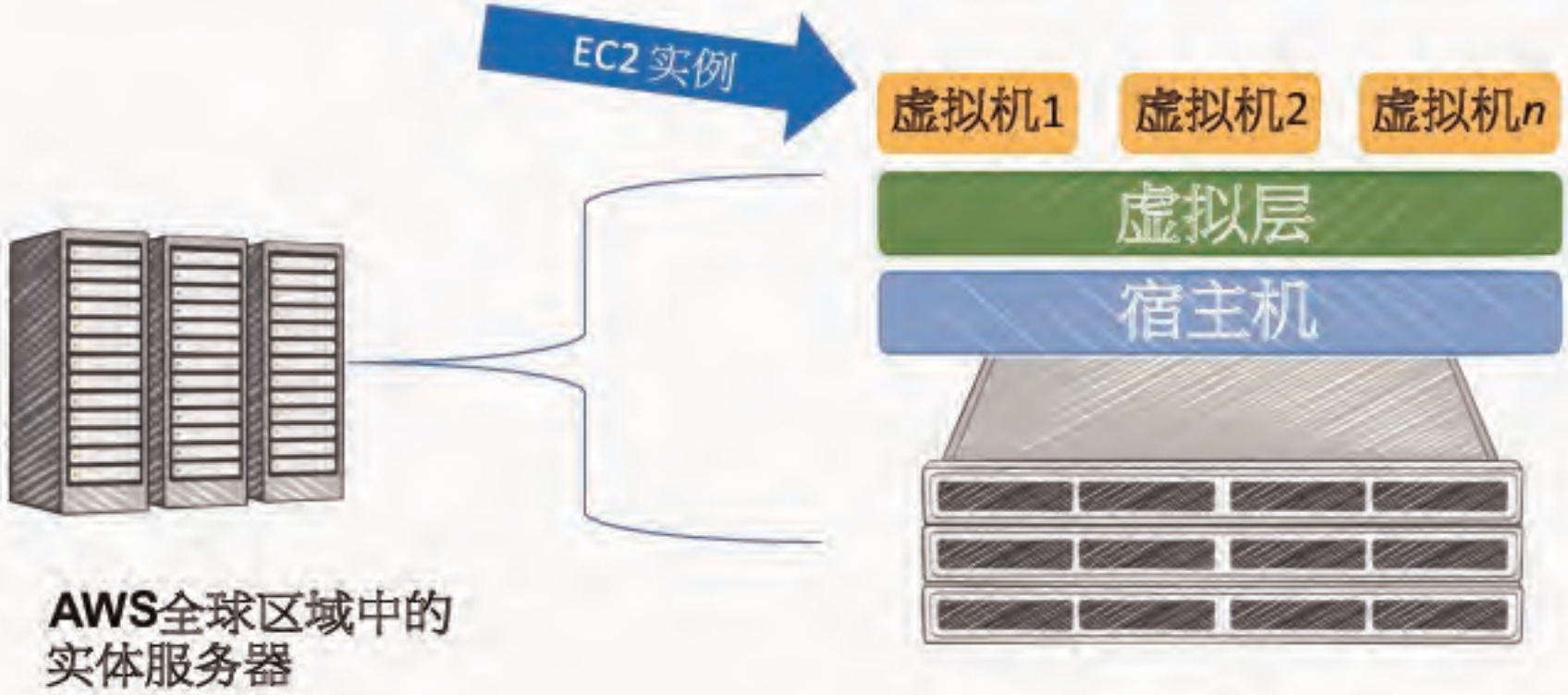
Source: Gartner (August 2016)

*Gartner, Magic Quadrant for Cloud Infrastructure as a Service, Worldwide, Leong, Lydia, Petri, Gregor, Gill, Bob, Dorosh, Mike, August 32016
This graphic was published by Gartner, Inc. as part of a larger research document and should be evaluated in the context of the entire document. The Gartner document is available upon request from AWS :
http://www.gartner.com/doc/reprints?id=1-2F2O5FC&ct=160616&st=sb

Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings or other designation. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

Amazon Elastic Compute Cloud (EC2)

在云端的弹性 虚拟服务器



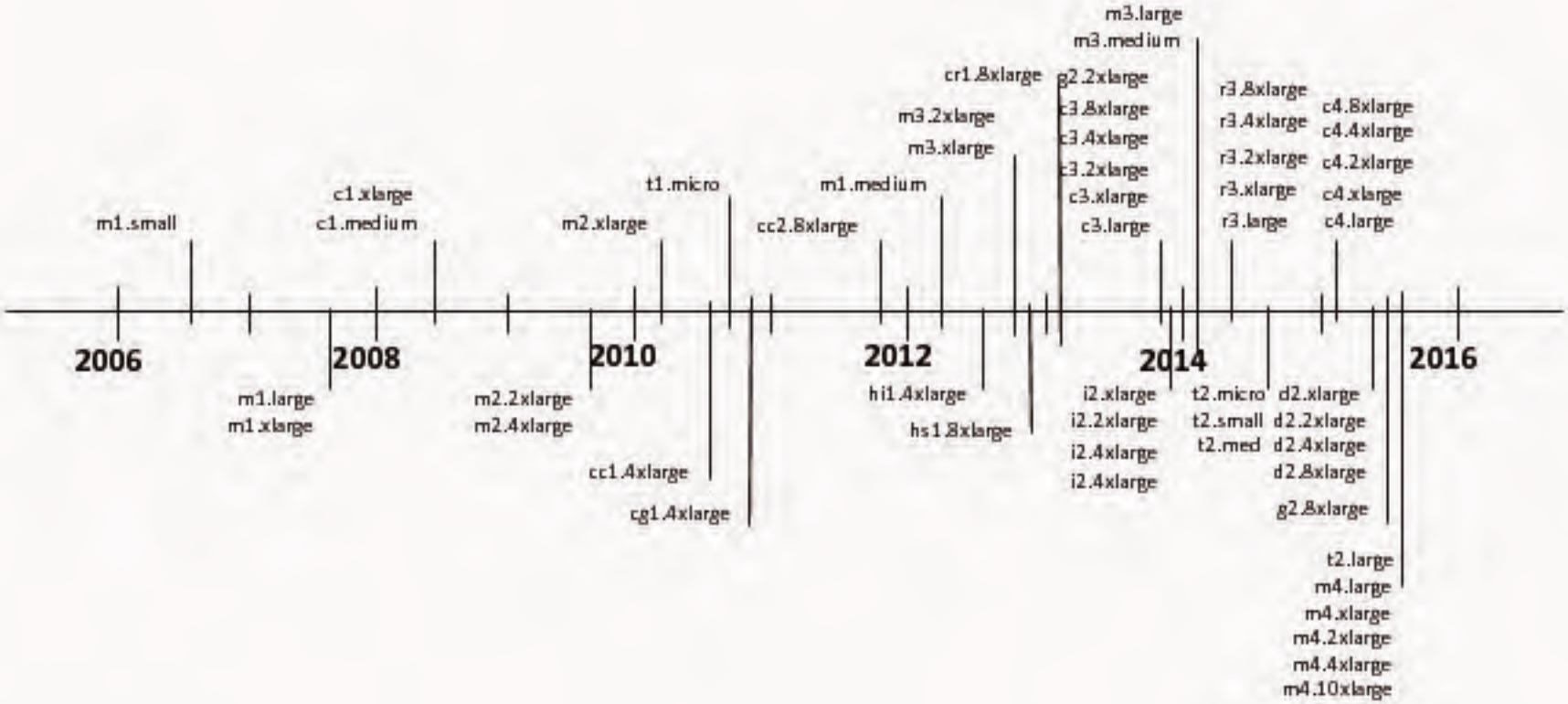
Amazon EC2 诞生于十年前...

APMCon

- 第一代, 仅有一种类型与大小
  - m1.small (1 vCPU, 1.7 GiB 内存, 160 GB 存储)
- 仅支持 Linux 操作系统
- 仅支持按需 (On-Demand) 付费的方式

Amazon EC2 实例的历史

APMCon



今天，丰富的操作系统选择

- 免费的 OS AMI
  - Amazon Linux、CentOS、FreeBSD、Ubuntu、Debian 等
- 付费 OS AMI (按小时付费)
  - Windows Server
  - Red Hat Enterprise Linux
  - SUSE Linux Enterprise
- Marketplace

EC2 实例: 家族与类型

APMCon

通用类型:	T2、M4、M3
计算优化:	C4、C3
内存优化:	X1、R3
GPU:	G2
存储优化:	
高 I/O:	I2
密集存储：	D2

Amazon 弹性计算云服务

APMCon

购买选项

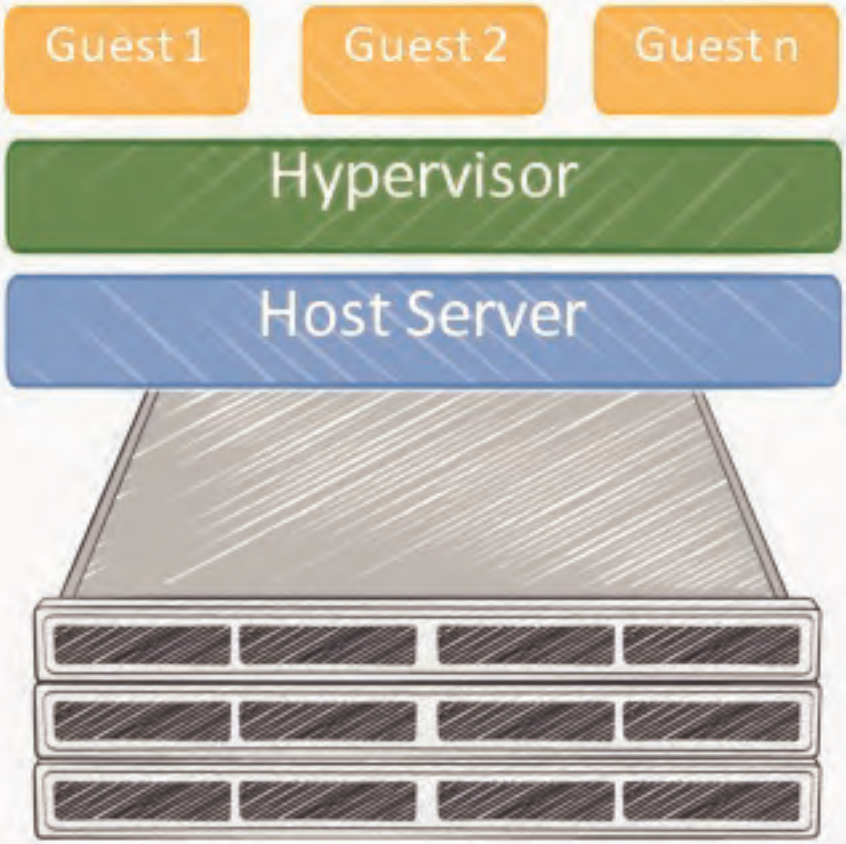
API

实例

网络

Amazon EC2 实例

APMCon



关于Amazon EC2 的性能

- 定义 **系统的性能** 和它在不同的工作负载下的特征
- Amazon EC2 实例的 **性能表现** 如何，以及如何提供灵活性和敏捷性
- 如何 **最大限度** 的利用好Amazon EC2 实例

关于 性能 的定义

获得一台服务器

APMCon

- 获取服务器用以完成所需的任务
- 基于不同的任务，性能的度量有很大的不同

性能的定义: 从问题的视角出发

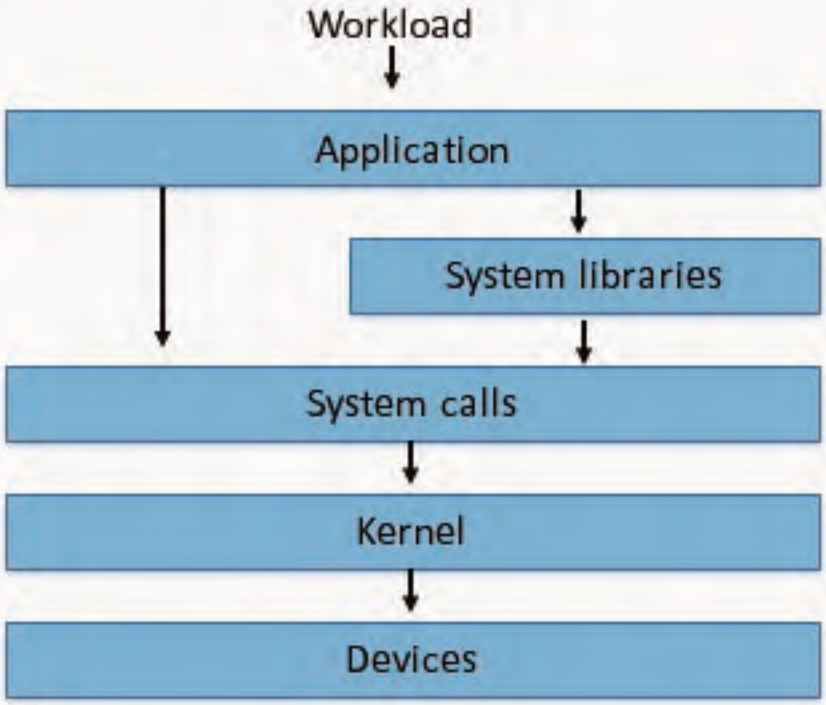
APMCon

性能的表现，取决于不同的视角:

响应时间

吞吐量

一致性



影响性能的因素

APMCon

资源	影响性能的因素	关键指标
CPU	CPU 插槽、CPU 核的数量、时钟频率, CPU bursting 能力	CPU 利用率、运行队列长度
内存	内存容量	空闲内存、匿名分页、线程交换
网络	最大带宽、包速率	接收吞吐量、最大带宽下传输吞吐量
磁盘	每秒钟完成的输入 / 输出操作，吞吐量	等待队列长度、设备利用率、设备错误

资源的利用率

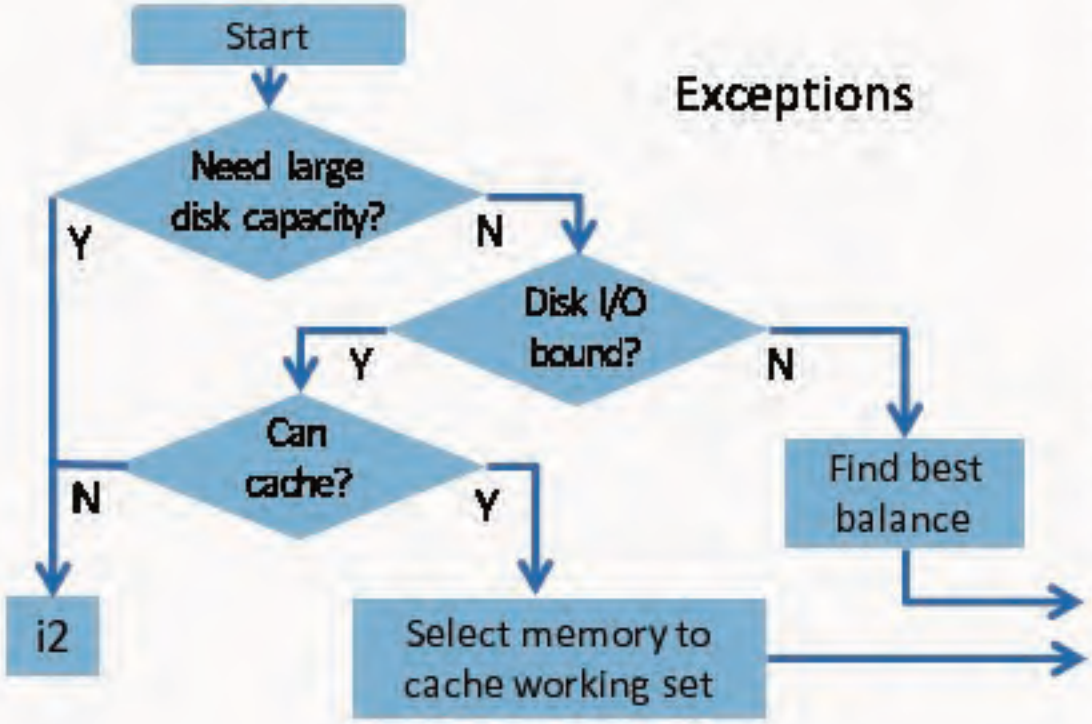
APMCon

- 对于给定的性能，有效的资源被使用的如何？
- 当处于在100%的利用率的情况下，则不能接受任何更多的工作
- 低利用率可以显示出购买的资源多于所需

实例选择 = 性能优化

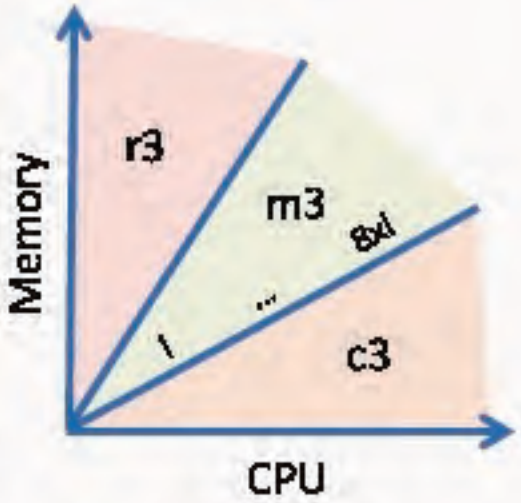
- 挑选一个适合的实例等同于资源的性能优化
- 获得新的实例前尽早的释放不需要的实例
- 找到一个理想的实例类型和工作负载的组合

实例选择流程图



Exceptions

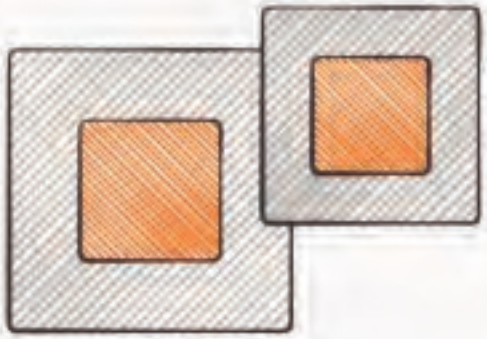
Trade-off



来源：Netflix

例子: Amazon EC2 C4 实例

- 最新一代的计算优化实例，最高性能的处理器和最高的性价比
- 定制的 Intel E5-2666 v3 at 2.9 GHz
- 可控制 P-state 和 C-state 配置

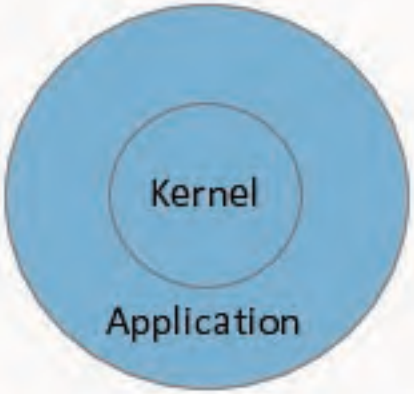


型号	vCPU	内存(GiB)	专用EBS带宽 (Mbps)
c4.large	2	3.75	500
c4.xlarge	4	7.5	750
c4.2xlarge	8	15	1,000
c4.4xlarge	16	30	2,000
c4.8xlarge	36	60	4,000

Amazon EC2 的 技术 特性

CPU 指令核保护等级

- CPU 有至少两个保护等级
- 不能在用户模式下执行特权指令来保护系统。应用程序利用系统调用访问内核



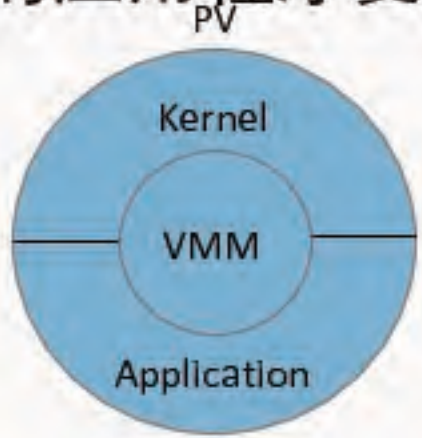
例子: Web 应用的系统调用

APMCon

```
[ec2-user@ip-10-0-121-0 ~]$ sudo strace -c -p 2440
Process 2440 attached
^CProcess 2440 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  0.00    0.000000           0       931        11 read
  0.00    0.000000           0       887           write
  0.00    0.000000           0       121           open
  0.00    0.000000           0       154           close
  0.00    0.000000           0      1357        32 stat
  0.00    0.000000           0       341           fstat
  0.00    0.000000           0        99        11 lstat
  0.00    0.000000           0       865           poll
  0.00    0.000000           0       121           mmap
  0.00    0.000000           0       121           munmap
  0.00    0.000000           0       220           brk
  0.00    0.000000           0        11           rt_sigaction
  0.00    0.000000           0        11           rt_sigprocmask
  0.00    0.000000           0        22           writev
  0.00    0.000000           0        66        22 access
```

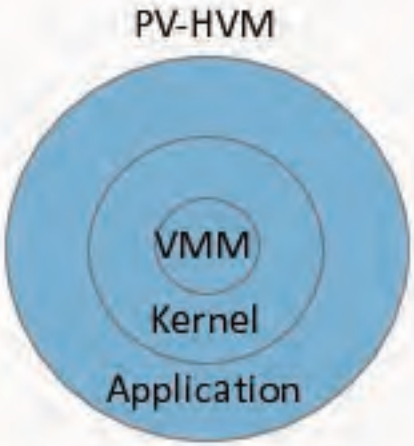
X86 CPU 虚拟化: 在 Intel VT-x 之前

- 二进制翻译特权指令
- 半虚拟化 (PV)
  - PV 需要穿透VMM，增加了延迟
  - 系统调用绑定的应用程序受到严重的影响



X86 CPU 虚拟化: 在 Intel VT-x 之后

- 硬件辅助的虚拟化 (HVM)
- PV-HVM 使用PV 驱动，巧妙的处理了那些较慢的模拟操作:
  - 例如：网络和块I/O



Xen 虚拟化模式

APMCon

- 性能最好的是最新的 Xen 所支持的 PV/HVM 混合模式
- 在 Amazon EC2上：
  - “**HVM**” == “**PVHVM**”，如果Linux kernel 支持
  - “**PV**” == “**PV**”
- 当前在Amazon EC2 上最快的模式就是: “HVM” (PVHVM)
- 未来最快的应该是 PVH

Xen 虚拟化模式

Optimal performance

Scope for improvement

Poor performance

P = paravirt.
VS = virt. in software, VH = virt. in hardware

	Type	Mode	With	Disk and Network	Interrupts, Timers	Emulated Motherboard, Legacy boot	Privileged Instructions and page tables
	Fully Virtualized	HVM		VS	VS	VS	VH
Old	Hybrid, Xen 3.0	HVM	PV drivers	P	VS	VS	VH
↓	Hybrid, Xen 4.0.1	HVM	PVHVM drivers	P	P	VS	VH
New	Hybrid, Xen 4.4	PV	HVM (PVH)	P	P	P	VH
	Fully Paravirtualized	PV		P	P	P	P

提示：使用PV-HVM AMIs 和 EBS

APMCon

区域	HVM (SSD) 支持 EBS 64 位	全虚拟化 实例存储 64 位	PV 支持 EBS 64 位	PV 实例存储 64 位	全虚拟化 (NAT) 支持 EBS 64 位	全虚拟化（即将） 支持 EBS 64 位
美国东部 弗吉尼亚北部	ami-8869aa05	ami-dc6aa9b1	ami-2a69aa47	ami-ed6aa980	ami-4868ab25	AWS Marketplace
美国西部 俄勒冈州	ami-7172b611	ami-fa76b29a	ami-7f77b31f	ami-9d75b1fd	ami-a275b1c2	AWS Marketplace
美国西部 加利福尼亚北部	ami-31490d51	ami-524b0f32	ami-a2490dc2	ami-15480c75	ami-004b0f60	AWS Marketplace
欧洲 爱尔兰	ami-f9dd458a	ami-e9d9419a	ami-4cdd453f	ami-42df4731	ami-a8dd45db	AWS Marketplace
欧洲 法兰克福	ami-ea26ce85	ami-e626ce89	ami-8527cf0a	ami-a125cdce	ami-5825cd37	AWS Marketplace
亚太区域 新加坡	ami-a59b49c6	ami-dd9e4cbe	ami-df9e4cbc	ami-96984af5	ami-a79b49c4	AWS Marketplace
亚太区域 首尔	ami-2b408b45	ami-d4408bba	无	无	ami-d14388bf	无
亚太区域 东京	ami-374db956	ami-ad41b5cc	ami-3e42b65f	ami-a940b4c8	ami-2443b745	AWS Marketplace
亚太区域 悉尼	ami-dc361ebf	ami-79361e1a	ami-63351d00	ami-95082bf6	ami-53371f30	AWS Marketplace
亚太区域 孟买	ami-ffbdd790	ami-b9b6dcd6	无	无	ami-e2b8d38d	无
南美洲 圣保罗	ami-6dd04501	ami-add144c1	ami-1ad34676	ami-8036bcec	ami-9336bcff	无
中国 北京	ami-8e6aa0e3	ami-1d6ba170	ami-77559f1a	ami-73569c1e	ami-7b549e16	无
AWS GovCloud	ami-8fc9770e	ami-6ec9770f	ami-d7ca74b6	ami-5fcb753e	ami-84cc72e5	无

关于 Timekeeping

APMCon

- 在一个实例中的 Timekeeping 很容易让人感到困惑
- gettimeofday(), clock_gettime(), QueryPerformanceCounter()
- TSC
  - CPU counter, accessible from userspace
  - Requires calibration, vDSO
  - Invariant on Sandy Bridge+ processors
- Xen 的 pvclock; 不支持 vDSO
- 在当前这一代的实例中，使用 TSC 作为时钟源（clocksource）

SR-IOV

- 单根 I/O 虚拟化 (SR-IOV)
  - PCIe 设备提供的虚拟化实例
- AWS “增强联网”
  - 适用于部分实例类型：C3、C4、R3、I2、M4、D2、X1
  - 仅仅用于 VPC 环境下
- 性能提升：
  - 提高网络吞吐量
  - 降低网络平均往返时延 (RTT) 和减少抖动
  - 提高取决于实例类型和网络

Linux 内核 调优

调优的目标

APMCon

1、	CPU Scheduler
2、	Virtual Memory
3、	Huge Pages
4、	File System
5、	Storage I/O
6、	Networking
7、	Hypervisor (Xen)

1. CPU Scheduler

APMCon

调整项:

- Scheduler class、priorities、migration latency、tasksets...

用途:

- 一些应用得益于通过使用 taskset(1)、numactl(8)、cgroups以及 优化 sched_migration_cost_ns 而减少进程迁移
- 一些 Java 应用得益于 SCHED_BATCH 减少上下文切换

```
# schedtool -B PID

# schedtool -R -p 20 <PIDs>
```

2. Virtual Memory

APMCon

调整项:

- swappiness、overcommit、OOM ...

用途:

- swappiness 设置为0用以禁用交换，停止文件页面缓存以释放更多的内存

```
vm.swappiness = 0                       # from 60

# Set the swappiness value as root
echo 10 > /proc/sys/vm/swappiness

# Alternatively, run this
sysctl -w vm.swappiness=10
```

3. Huge Pages

APMCon

内存页面尺寸为2M 或者 4M，而不是常见的 4K。降低各种CPU开销并且提高MMU 页面到缓存的转换

调整项:

- 显性的使用huge page、transparent huge pages (THPs)

用途:

- THPs (内核中启用)，依赖于工作负载和CPU，有时候可以提高性能 (~5% CPU负荷), 但是有时候会影响性能 (~25% CPU 负荷，当 %usr, 以及 %sys refrag)。禁用的方法：

```
# grep Huge /proc/meminfo
...
Hugepagesize: 2048 kB
...

# echo never > /sys/kernel/mm/transparent_hugepage/enabled    # from madvise

# java -XX:+UseLargePages
```

4. File System

APMCon

调整项:

- 页面缓存的刷新，文件系统的类型和可调整的参数 (例如：ZFS)

用途:

- 页面缓存刷新的行为可以调整为: background flush earlier、aggressive flush later
- 禁止访问时间戳和其它的选项，依赖于具体的文件系统

```
vm.dirty_ratio = 80                          # from 40
vm.dirty_background_ratio = 5                # from 10
vm.dirty_expire_centisecs = 12000            # from 3000
mount -o defaults,noatime,discard,nobarrier …
```

5. Storage I/O

APMCon

调整项:

- Read ahead size、number of in-flight requests、I/O scheduler、volume stripe width...

用途:

- 一些应用（例如 Cassandra）对read ahead size 非常敏感
- SSDs 可以使用更好的 “noop” scheduler (非缺省的设置)
- 调整chunk size 和 stripe width 以匹配任务负载

```
/sys/block/*/queue/rq_affinity 2
/sys/block/*/queue/scheduler       noop
/sys/block/*/queue/nr_requests 256
/sys/block/*/queue/read_ahead_kb   256
mdadm -chunk=64 ...
```

6. Networking

APMCon

调整项:

- TCP buffer sizes、TCP backlog、device backlog、TCP reuse ...

用途:

```
net.core.somaxconn = 1000
net.core.netdev_max_backlog = 5000
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_wmem = 4096 12582912 16777216
net.ipv4.tcp_rmem = 4096 12582912 16777216
net.ipv4.tcp_max_syn_backlog = 8096
net.ipv4.tcp_slow_start_after_idle = 0
net.ipv4.tcp_tw_reuse = 1
net.ipv4.ip_local_port_range = 10240 65535
net.ipv4.tcp_abort_on_overflow = 1    # maybe
```

7. Hypervisor (Xen)

APMCon

调整项:

- PV/HVM (AMI 提供)
- 内核 clocksource，从慢到快: hpet、xen、tsc

用途:

- 调优clocksource ，设置为TSC (小心时钟漂移)。较好的情况下 CPU 用量减少 30%, 并且应用程序的平均延迟降低了 43%

```
# cat  /sys/devices/system/clocksource/clocksource0/current_clocksource
xen

#echo tsc >
/sys/devices/system/clocksource/clocksource0/current_clocksource
```

提示: 使用TSC 作为时钟源

APMCon

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

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

Change with:

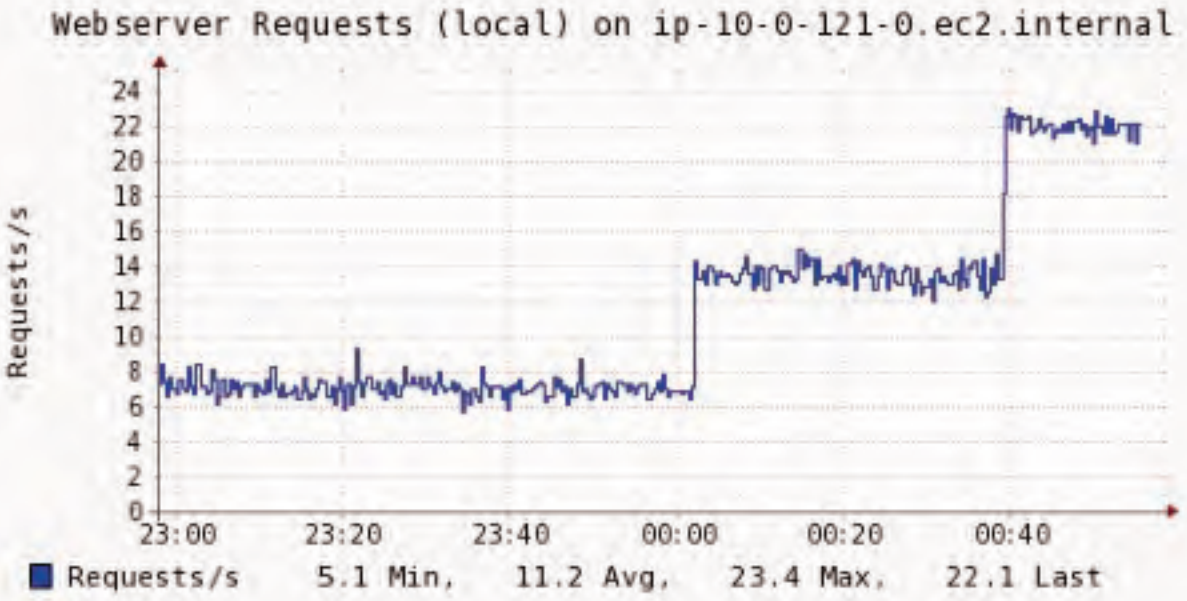
```
# emacs /boot/grub/menu.list && reboot

# cat /proc/cmdline
root=/dev/xvda1 ro clock source=tsc
```

性能 监测

例子：Web 应用

APMCon

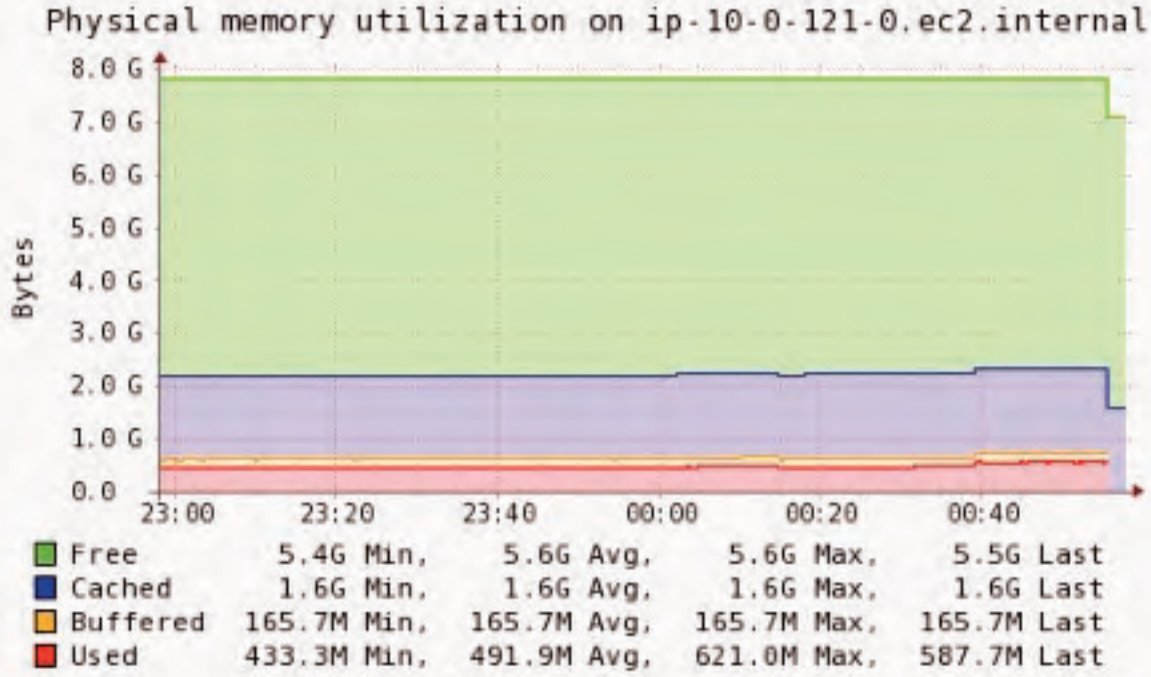
- 在Apache 上安装 MediaWik，并带有140页的内容
- 负载随时间间隔而增加



例子：Web 应用

APMCon

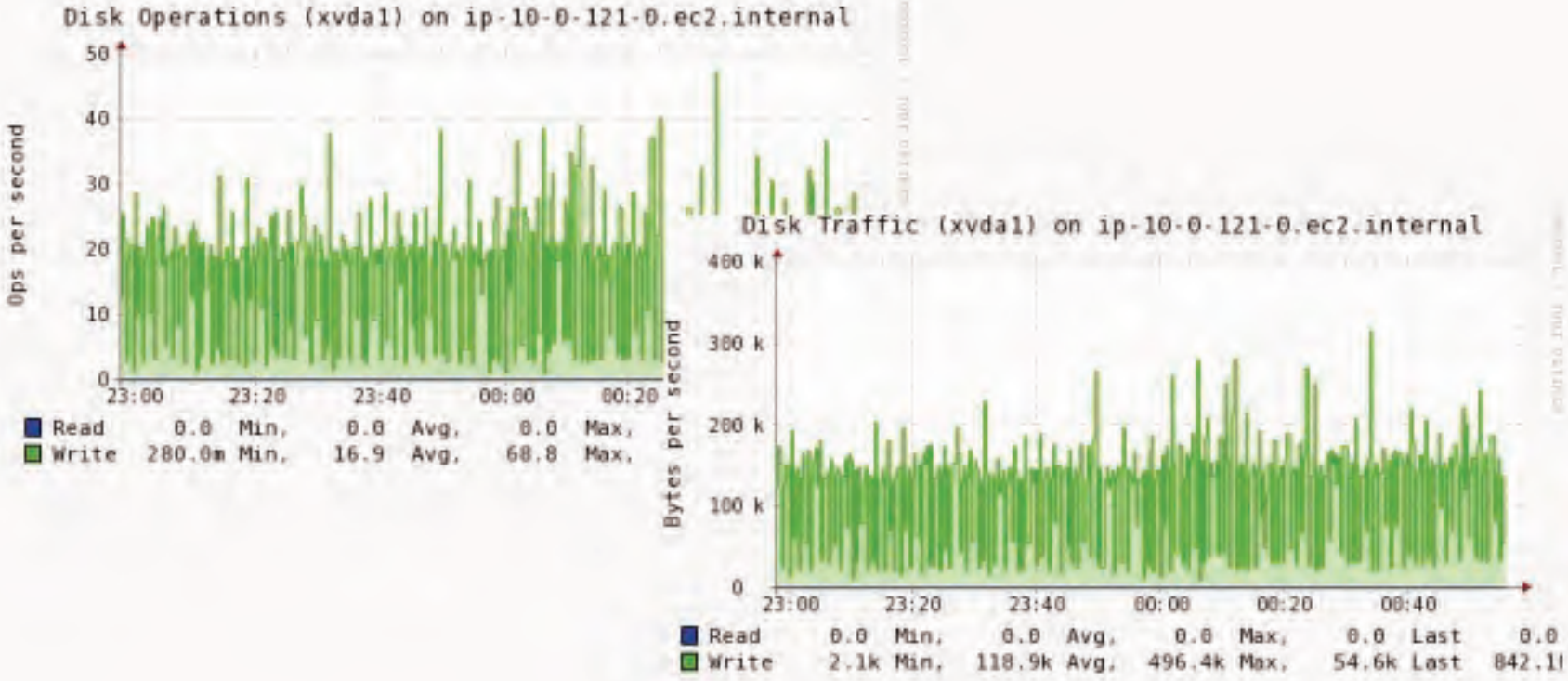
- 内存使用统计



例子：Web 应用

APMCon

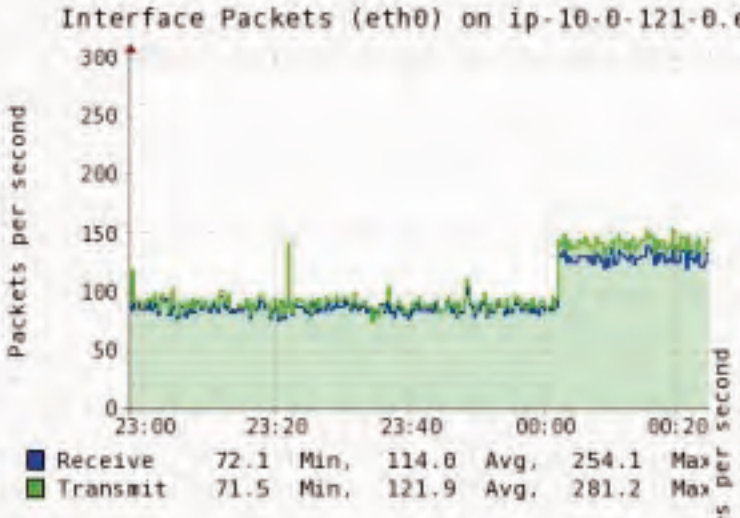
- 磁盘使用统计

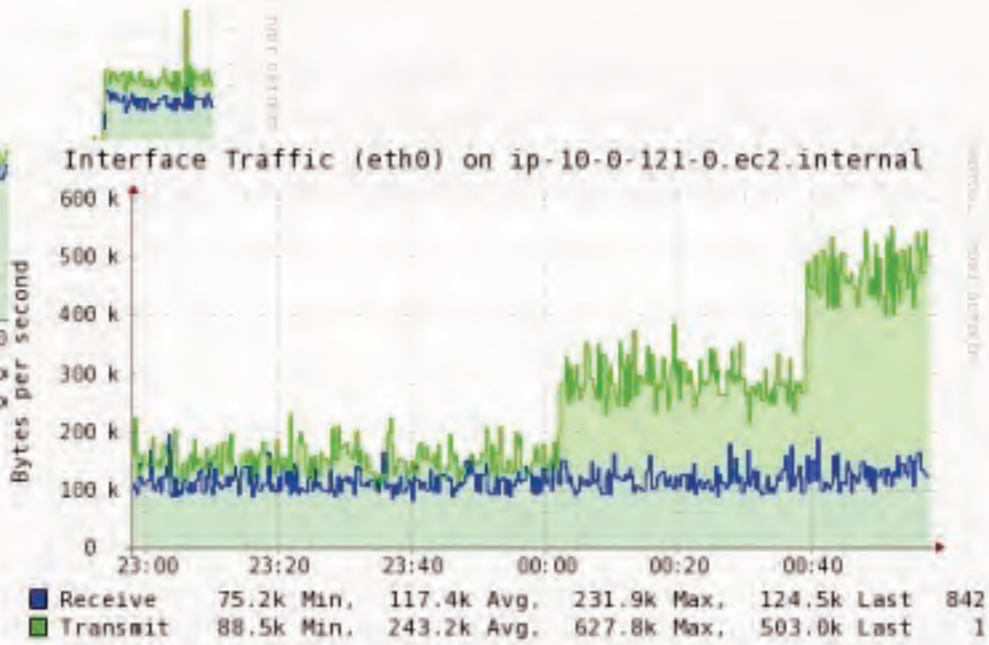


例子：Web 应用

APMCon

- 网络使用统计

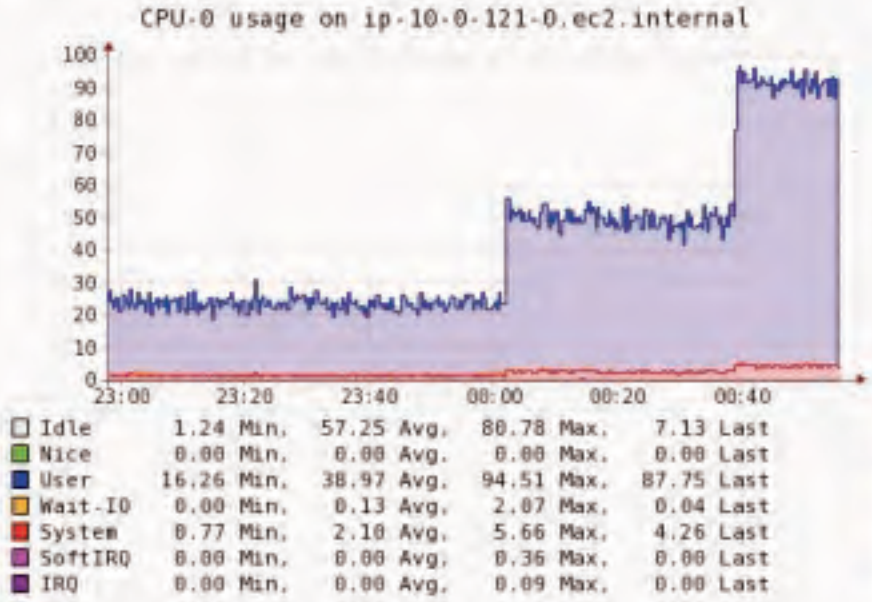


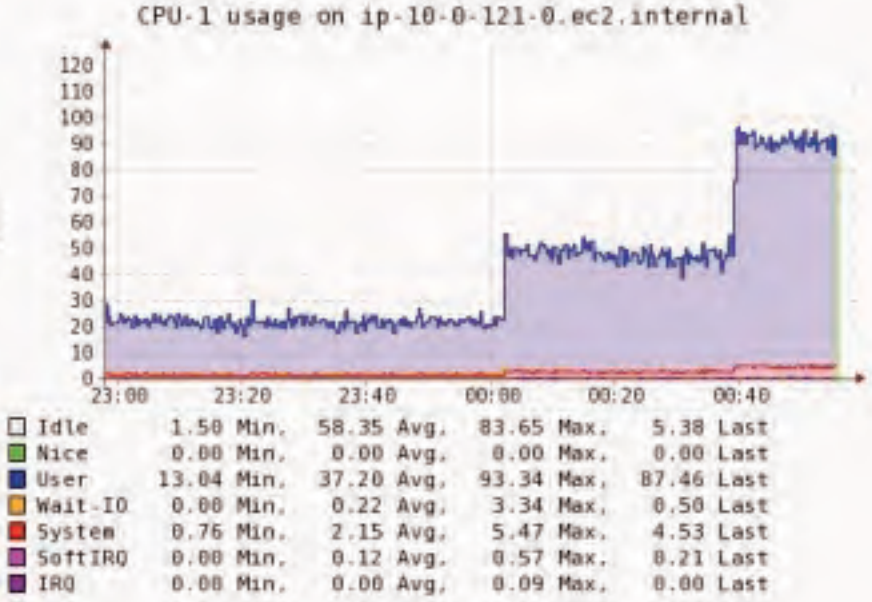


例子：Web 应用

APMCon

- CPU 使用统计





无效的调优

APMCon

“打地鼠”式的优化:

- 随机的调整参数直到问题消失

“街灯”式的优化:

- 1. 选择监控工具
  - 只使用自己熟悉的
  - 从互联网找工具
  - 随机选择工具
- 2. 运行工具
- 3. 观察问题

有效调优

基于观察，数据驱动:

1. 对系统负荷以及资源使用情况进行观察
2. 分析结果:
   - 确定使用的硬件 / 软件的组件
   - 研究组件的可调整参数
   - 量化预期的改进
3. 调优，实验以及确定目标

USE 方法

APMCon

USE 方法 -

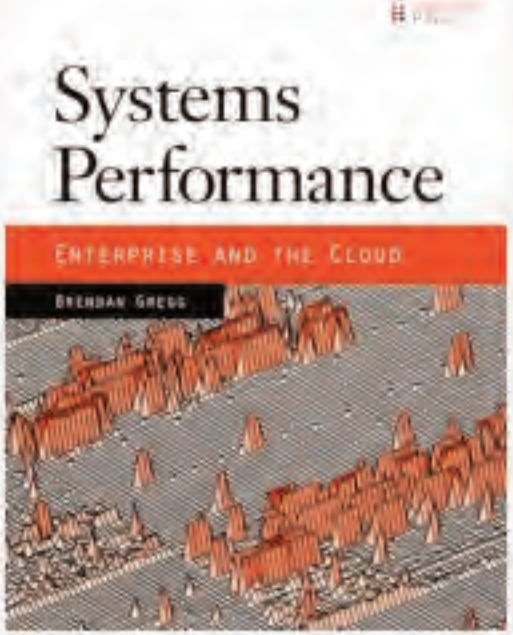
对每一项硬件和软件资源，检查它们的：

1. 利用率（**U**tilization）
2. 饱和度（**S**aturation）
3. 错误数（**E**rrors）

如果，资源约束显示为高饱和度或者高利用率

- 调整或者改变实例类型
- 了解资源的可调参数

USE 方法帮我们发现问题，然后使用工具解决问题



实例之外的考虑

APMCon

实例免责

- 复杂的性能问题没有明显的原因，也许是实例的问题？
- 有时候分析之后确认与实例有关。但是大多数与应用有关 (80/20 原则)

80/20 原则:

- 80%：通过提高应用程序的重构和优化获得改进
- 20%：OS 调优、实例或者基础架构的改善

20/80 延迟异常原则:

- 20%:：延迟异常是由程序代码引起
- 80%：是由实例、 crontab、网络以及 JVM GC 等引起

分析的视角

APMCon

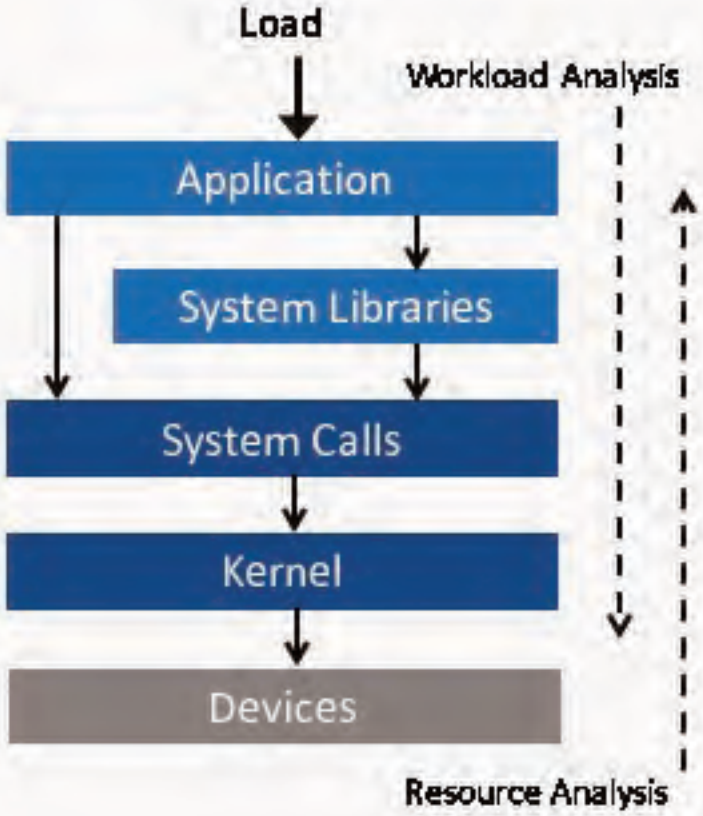
工作负载分析

- “自顶向下”分析
- 从应用的附载着手
  然后，分解请求时间

资源分析：

- “自底向上”分析
- 从资源的性能开始着手，然后是工作负载

例如：USE 方法应用于负载的特征上



Linux 性能工具

APMCon

1. Statistical 工具
2. Profiling 工具
3. Tracing 工具
4. Hardware 计数器

1. Statistical 工具

APMCon

最常用的有 vmstat、pidstat、sar 等

```
$ sar -n TCP,ETCP,DEV 1
Linux 3.2.55 (test-e4f1a80b)    08/18/2014    _x86_64_ (8 CPU)

09:10:43 PM  IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s rxcmp/s txcmp/s  rxmcst/s
09:10:44 PM     lo     14.00     14.00      1.34      1.34    0.00    0.00      0.00
09:10:44 PM   eth0   4114.00   4186.00   4537.46  28513.24    0.00    0.00      0.00

09:10:43 PM  active/s passive/s    iseg/s    oseg/s
09:10:44 PM     21.00      4.00   4107.00  22511.00

09:10:43 PM  atmptf/s  estres/s retrans/s isegerr/s   orsts/s
09:10:44 PM      0.00      0.00     36.00      0.00      1.00
[…]
```

2. Profiling 工具

APMCon

Profiling: 典型的应用场景

- 对 CPU 堆栈跟踪的采样，可以解释 CPU 的使用
- 内存对象的频度计数，可以解释内存的使用

Profiling 产生的调优

- 热代码 (Hot code) 的路径 -> 相关的可调参数 / 配置？
- 频繁的对象分配 -> 避免这种情况的方法？

Profiling 类型

APMCon

程序的profiling

- 取决于应用和开发语言
- 例如： Java Flight Recorder, Yourkit, Lightweight Java Profiler
- 许多工具缺乏准确以及存在问题；测试、验证、交叉检查

系统的 profiling

- Linux perf_events (“perf” 命令)
- ftrace 可以用作内核函数的计数
- SystemTap 拥有各种分析功能

面向应用的 Profiling: LJP

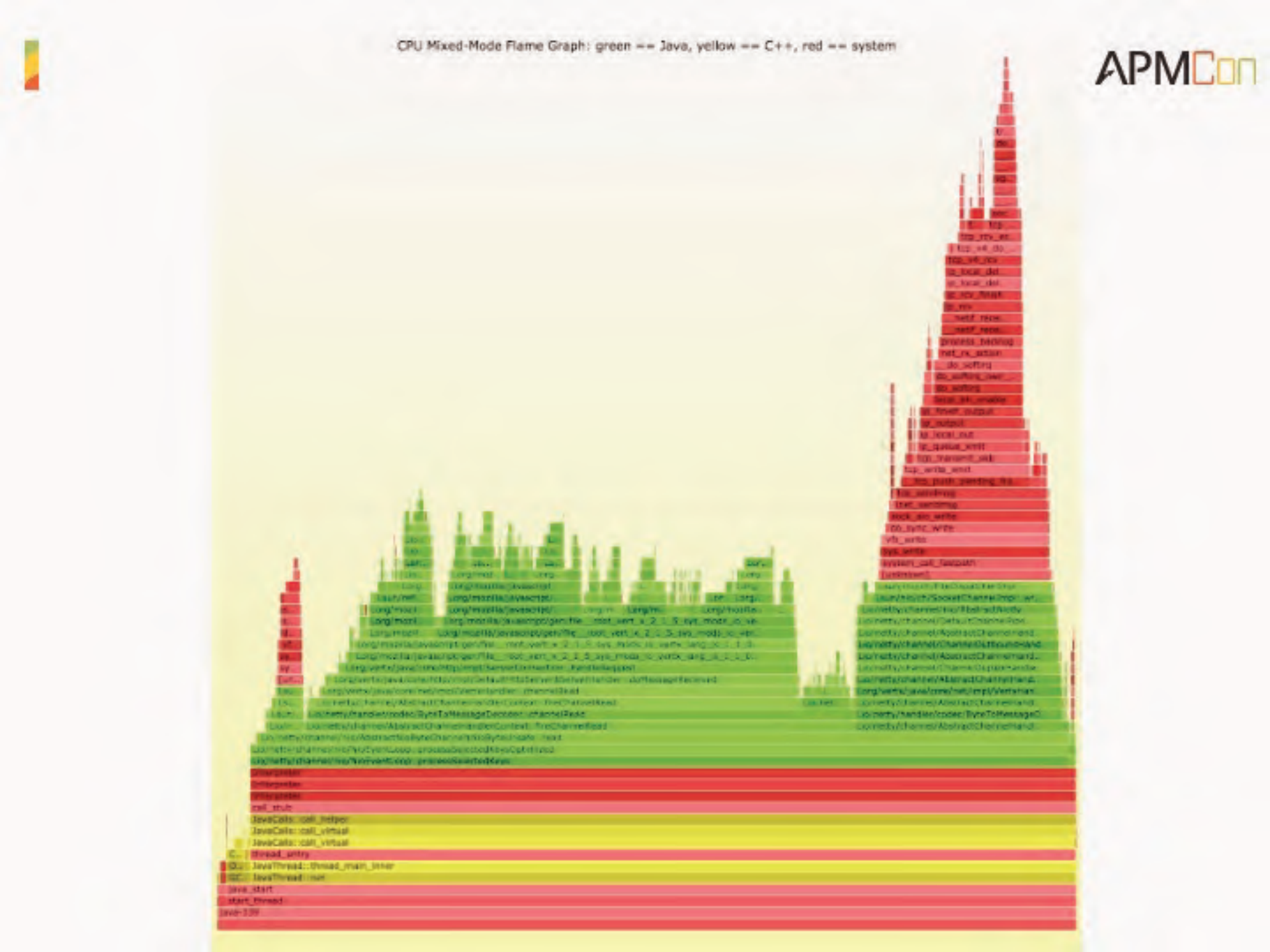
APMCon

轻量级的 Java Profiler (LJP)

- 开源的、免费的、异步的 CPU profiler
- 使用代理 (agent) 转储 hprof 同样格式的输出
  - https://code.google.com/archive/p/lightweight-java-profiler/

profiling 的输出结果可以输出为 -

火焰图 (flame graphs)


CPU Mixed-Mode Flame Graph: green == Java, yellow == C++, red == system
APMCon

系统Profiling: perf_events

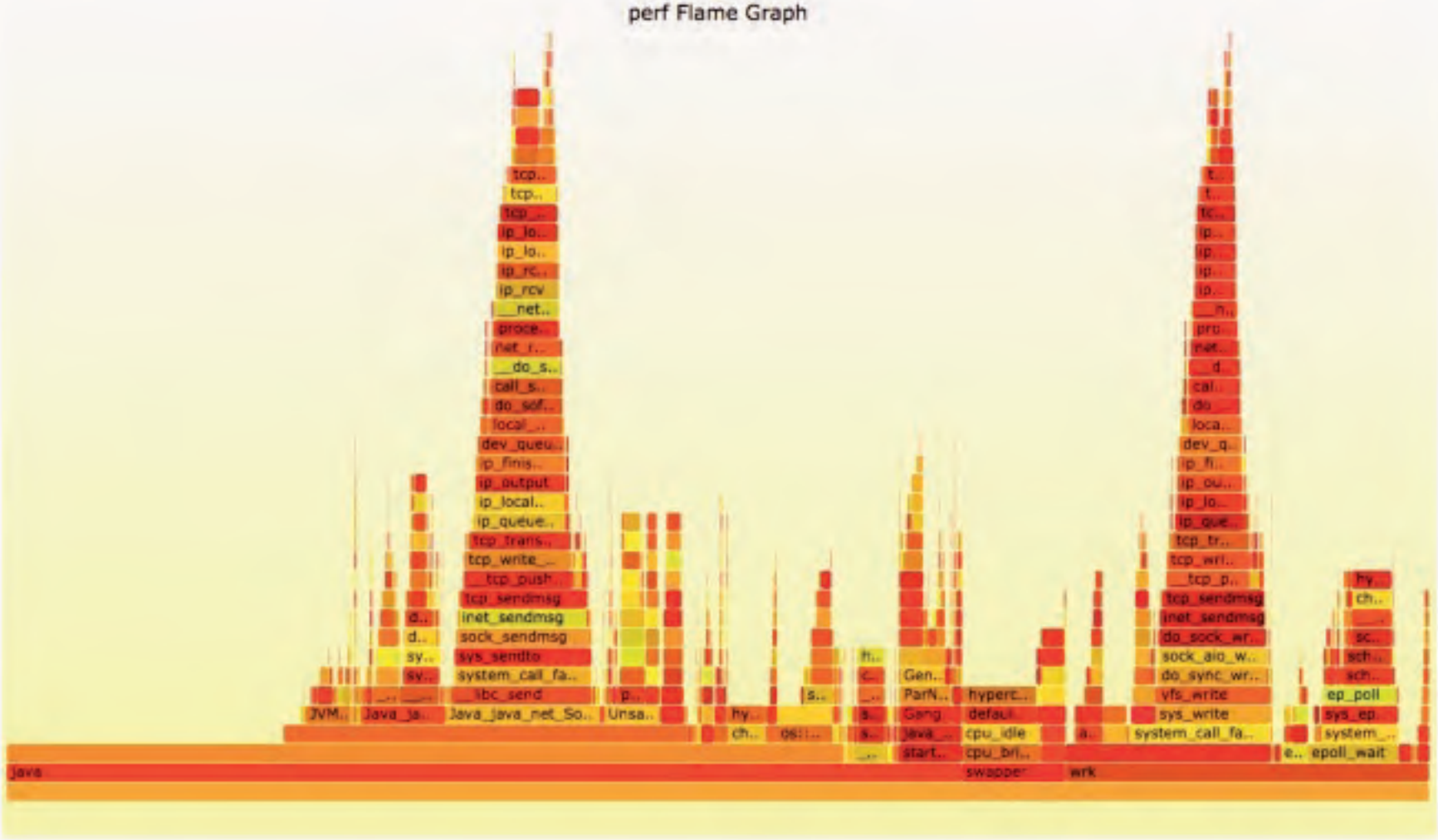
APMCon

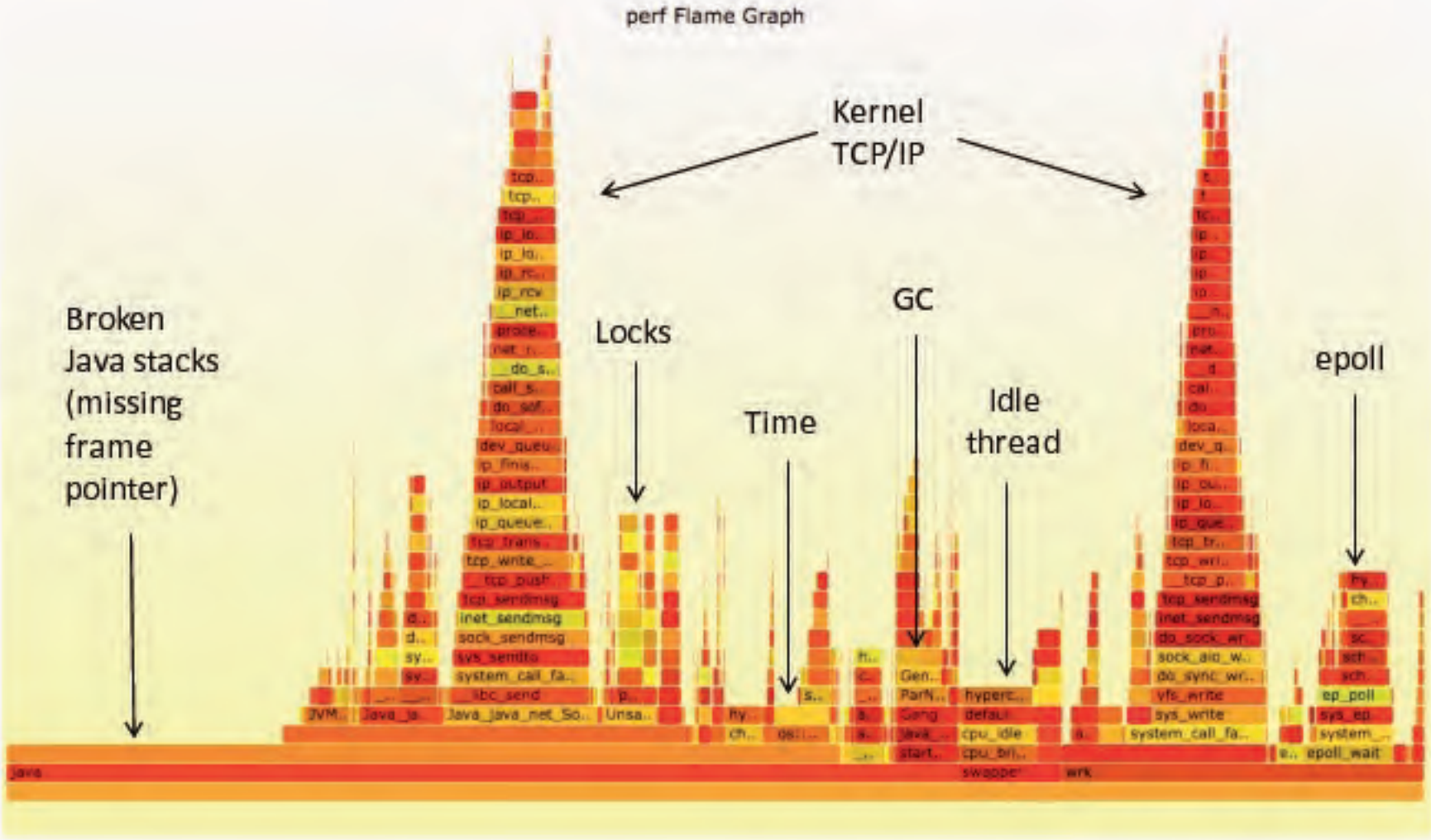
perf_events 对CPU 堆栈跟踪进行采样，可以显示为：

- 应用的逻辑
  - 依赖于应用和 VM，可以展示出高级的逻辑关系
- JVM 内部信息和库
- Linux 内核

perf CPU 火焰图 (flame graphs)：

```
# git clone https://github.com/brendangregg/FlameGraph
# cd FlameGraph
# perf record -F 99 -ag -- sleep 60
# perf script | ./stackcollapse-perf.pl | ./flamegraph.pl > perf.svg
```


perf Flame Graph
tcp..
tcp..
tcp_...
ip_lo..
ip_lo..
ip_rc..
ip_rcv
__net..
proce..
net_r..
__do_s..
call_s..
do_sof..
local_..
dev_queu..
ip_finis..
ip_output
ip_local..
ip_queue..
tcp_trans..
tcp_write_..
__tcp_push
tcp_sendmsg
d..
inet_sendmsg
d..
sock_sendmsg
sy..
sys_sendto
sy..
system_call_fa..
__libc_send
JVM..
Java_ja..
Java_java_net_So..
Unsa..
hy..
ch..
os::...
fi..
c..
s..
_...
s..
s..
_...
Gen..
ParN..
Gang
java_...
start..
hyperc..
defaul..
cpu_idle
cpu_bri..
swapper
a..
wrk
t..
t..
tc..
ip..
ip..
ip..
ip..
__n..
pro..
net..
__d..
cal..
do_..
loca..
dev_q..
ip_fi..
ip_ou..
ip_lo..
ip_que..
tcp_tr..
tcp_wri..
__tcp_p..
tcp_sendmsg
inet_sendmsg
do_sock_wr..
sock_aio_w..
do_sync_wr..
vfs_write
sys_write
system_call_fa..
hy..
ch..
__..
sc..
sch..
sch..
ep_poll
sys_ep..
system_..
e..
epoll_wait
java


perf Flame Graph
Kernel
TCP/IP
Broken
Java stacks
(missing
frame
pointer)
Locks
Time
GC
Idle
thread
epoll
java
swapper
wrk

3. Tracing 工具

APMCon

Linux 中现有的系统 tracers工具 -

1. ftrace
2. perf_events
3. eBPF
4. SystemTap
5. ktap
6. LTTng
7. dtrace4linux
8. Oracle Linux DTrace
9. Sysdig

接下来，归纳一下 ftrace & perf_events ...

Linux 内核的一部分

- 第一次出现是在 2.6.27 (2008)，在后续的版本中得到增强
- 直接使用 /sys/kernel/debug/tracing

前端工具: perf-tools 集合

- https://github.com/brendangregg/perf-tools

ftrace tool: iosnoop

APMCon

```
# ./iosnoop -ts
Tracing block I/O. Ctrl-C to end.
STARTs          ENDs            COMM       PID   TYPE DEV    BLOCK     BYTES LATms
5982800.302061 5982800.302679 supervise  1809  W    202,1  17039600  4096   0.62
5982800.302423 5982800.302842 supervise  1809  W    202,1  17039608  4096   0.42
5982800.304962 5982800.305446 supervise  1801  W    202,1  17039616  4096   0.48
5982800.305250 5982800.305676 supervise  1801  W    202,1  17039624  4096   0.43
[…]
```

```
# ./iosnoop -h
USAGE: iosnoop [-hQst] [-d device] [-i iotype] [-p PID] [-n name] [duration]
                 -d device       # device string (eg, "202,1)
                 -i iotype       # match type (eg, '*R*' for all reads)
                 -n name         # process name to match on I/O issue
                 -p PID          # PID to match on I/O issue
                 -Q              # include queueing time in LATms
                 -s              # include start time of I/O (s)
                 -t              # include completion time of I/O (s)
[…]
```

perf_events

APMCon

Linux 代码的一部分

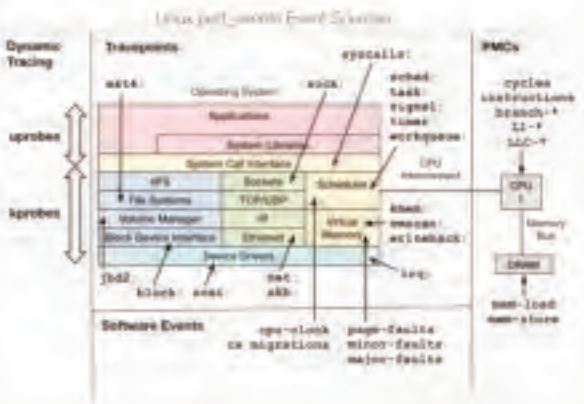
- 位于tools/perf 之下

强大的工具集合分析器

- interval sampling, CPU performance counter events
- user and kernel dynamic tracing
- kernel line tracing and local variables (debuginfo)
- kernel filtering,in-kernel 计数器 (perf stat)

高级用途需要内核 debuginfo

- 通常要大于100M 字节



perf_events 示例

APMCon

```
# perf record -e skb:consume_skb -ag -- sleep 10
# perf report
[...]
    74.42%  swapper  [kernel.kallsyms]  [k] consume_skb
            |
            --- consume_skb
                arp_process
                arp_rcv
                __netif_receive_skb_core
                __netif_receive_skb
                netif_receive_skb
                virtnet_poll
                net_rx_action
                __do_softirq
                irq_exit
                do_IRQ
                ret_from_intr
[…]
```

← Summarizing stack traces for a tracepoint

perf_events can do many things,
it is hard to pick just one example

4. Hardware 计数器

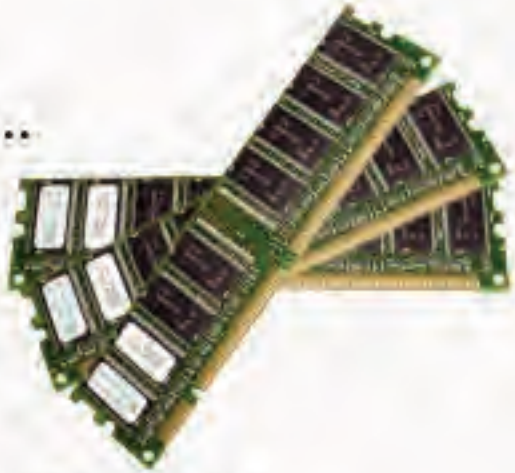
APMCon

Model Specific Registers (MSRs)

- 基本信息： timestamp clock, temperature, power
- 部分已经存在于 Amazon EC2

性能监控计数器 (PMCs)

- 高级信息：cycles, stall cycles, cache misses...
- 缺省情况下Amazon EC2 不包含



周期性CPU 用量的根本原因

- 例如：更高的内存用量导致更多的延迟周期 (stall cycles)

MSRs （ Model Specific Registers ）

APMCon

可以用来验证CPU 的真实频率

- 能够适应 turboboost，很重要的实现性能对比
- 项目地址

https://github.com/brendangregg/msr-cloud-tools

Real CPU MHz

```
ec2-guest# ./showboost
CPU MHz      : 2500
Turbo MHz    : 2900 (10 active)
Turbo Ratio  : 116% (10 active)
CPU 0 summary every 5 seconds...

TIME         C0_MCYC         C0_ACYC         UTIL   RATIO    MHz
06:11:35     6428553166      7457384521       51%    116%    2900
06:11:40     6349881107      7365764152       50%    115%    2899
06:11:45     6240610655      7239046277       49%    115%    2899
[...]
```

总结

- 实例的 选择 = 性能 优化
- 针对Linux 内核的调优
- 性能 监测

参考链接

APMCon

Amazon EC2 -

- http://aws.amazon.com/ec2/instance-types/
- https://docs.aws.amazon.com/zh_cn/AWSEC2/latest/UserGuide/instance-types.html
- https://docs.aws.amazon.com/zh_cn/AWSEC2/latest/UserGuide/enhanced-networking.html
- https://docs.aws.amazon.com/zh_cn/AWSEC2/latest/UserGuide/concepts.html

AWS SUMMIT beijing

AWS技术峰会2016·北京

2016年9月7日 – 9月8日

北京国际饭店会议中心

参会注册， http://www.awssummit.cn/Register/

THANK YOU