AWS

SUMMIT

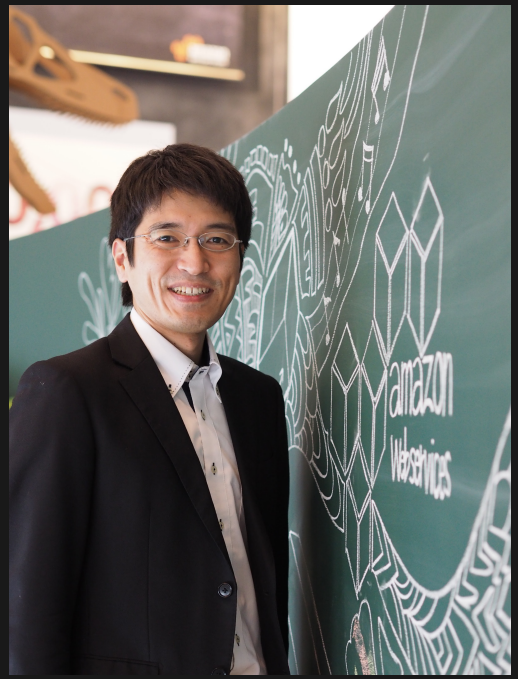
Amazon EC2 Performance Deep Dive

Takashi Ogawa / Solutions Architect (HPC/CAE)
Amazon Web Services Japan K.K.

June 2, 2017

amazon web services

自己紹介



名前：小川 貴士 (おがわ たかし)

所属：アマゾンウェブサービスジャパン
　　ソリューションアーキテクト

担当エリア： CAEを中心としたHPCのお客様

経歴： SIerでCAE/HPCのインフラエンジニア

好きなAWSサービス：Amazon EC2、AWS Batch

このセッションの目的 ~Amazon EC2 Performance Deep Dive~

- ワークロードに合った適切なEC2インスタンスを選択できるようにする
- コンピューティング/ストレージ/ネットワークに対して最適な性能を得る設定を理解する
- 最適な性能を得る事で余計なリソース利用を防ぐ
- パフォーマンス問題に遭遇した際の、対処アイテムを増やす

※本資料は後日公開予定です、何かのお役に立てば幸いです。

アジェンダ

- EC2インスタンスリソースよる性能要素
- I/O関連のパフォーマンスオプション
- パフォーマンスクレジット
- その他のパフォーマンス要素
- まとめ

アジェンダ

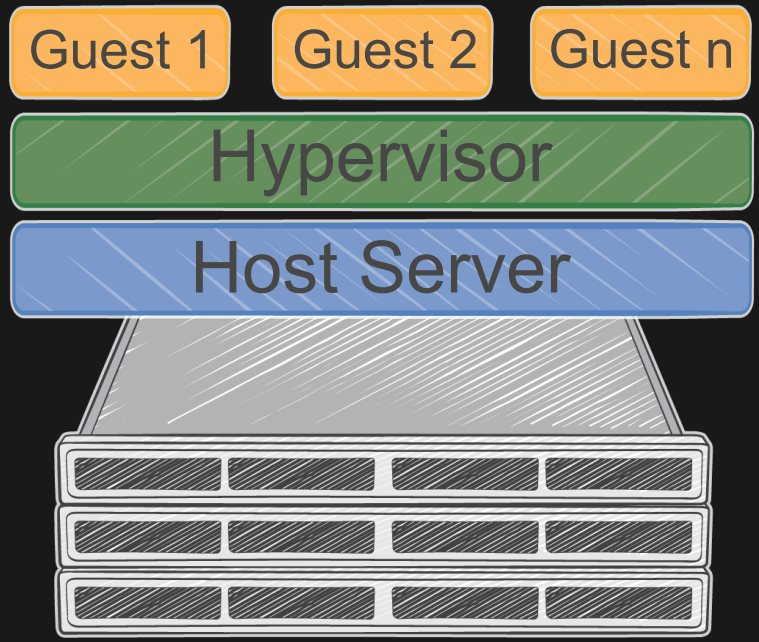
- EC2インスタンスリソースよる性能要素
- I/O関連のパフォーマンスオプション
- パフォーマンスクレジット
- その他のパフォーマンス要素
- まとめ

の順で物理から徐々に
レイヤーを上げてお話します

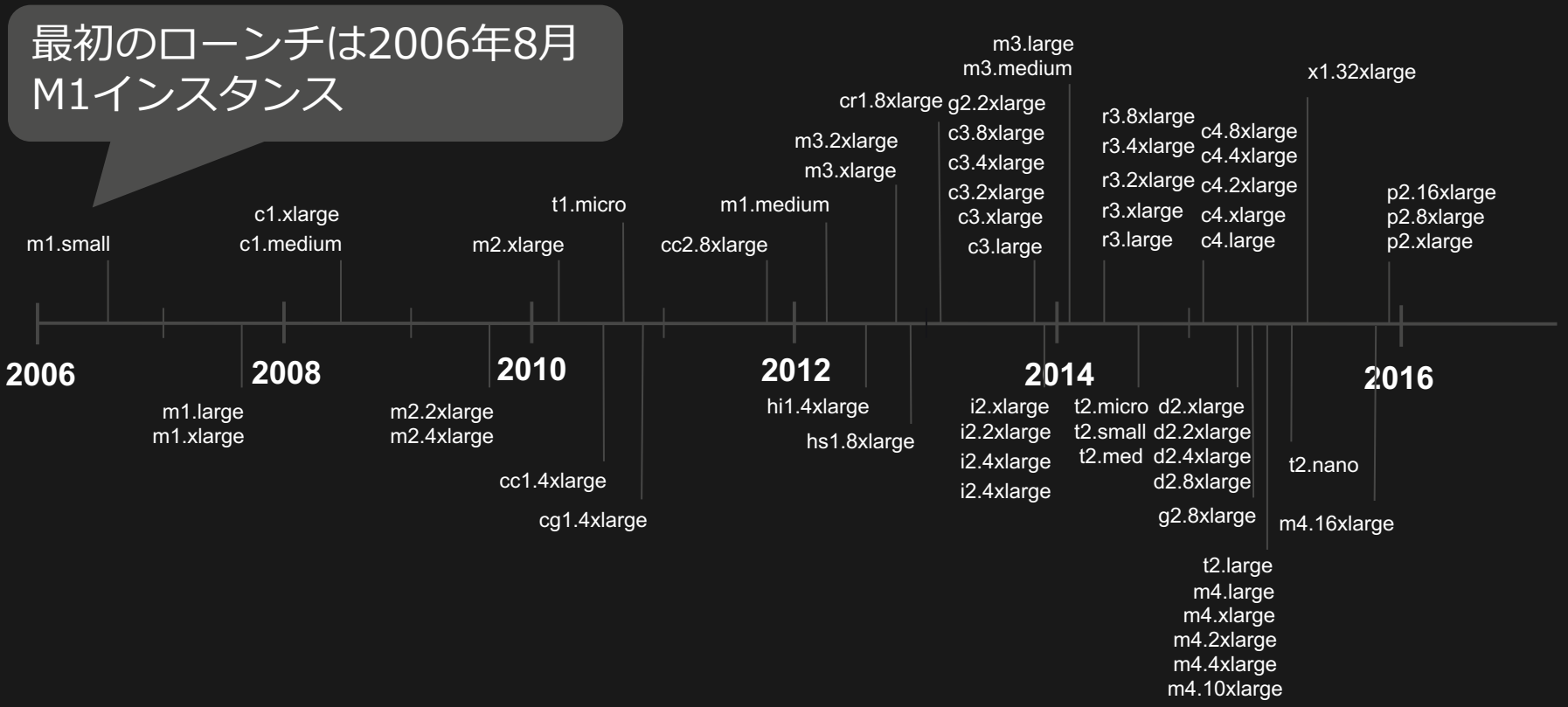
アジェンダ

- EC2インスタンスリソースよる性能要素
- I/O関連のパフォーマンスオプション
- パフォーマンスクレジット
- その他のパフォーマンス要素
- まとめ

Amazon EC2 インスタンス

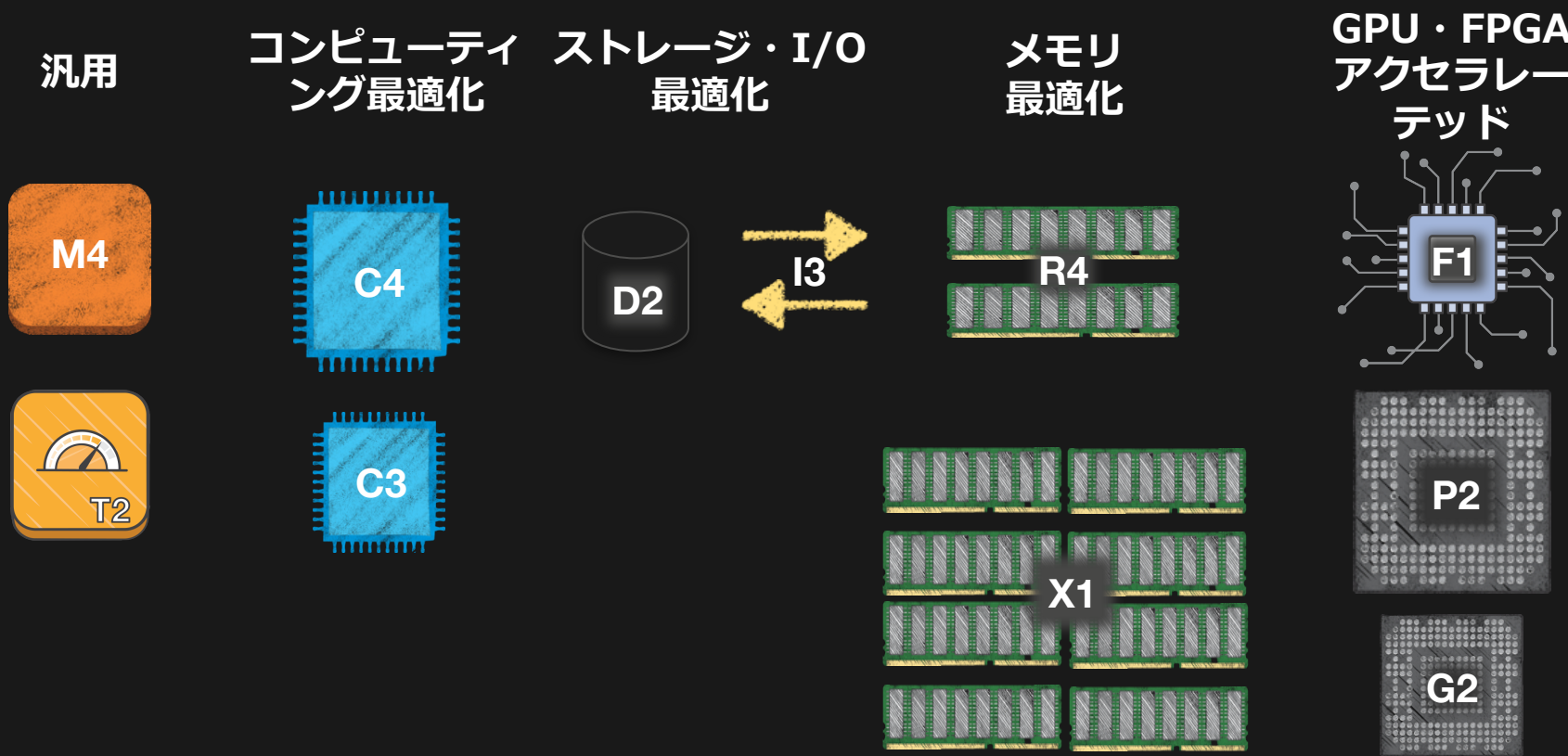


EC2インスタンスの歴史



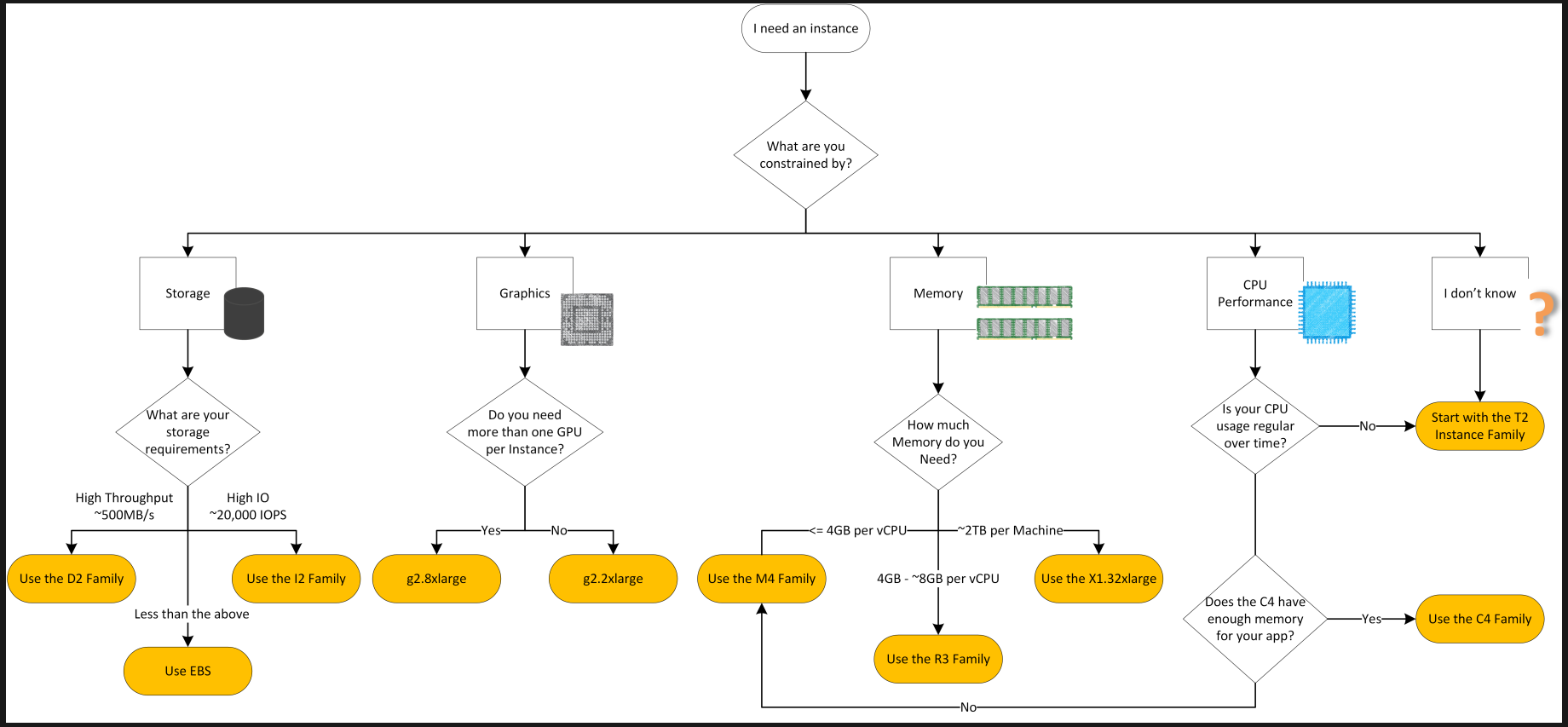
EC2インスタンスのネーミングポリシー

EC2インスタンスファミリー



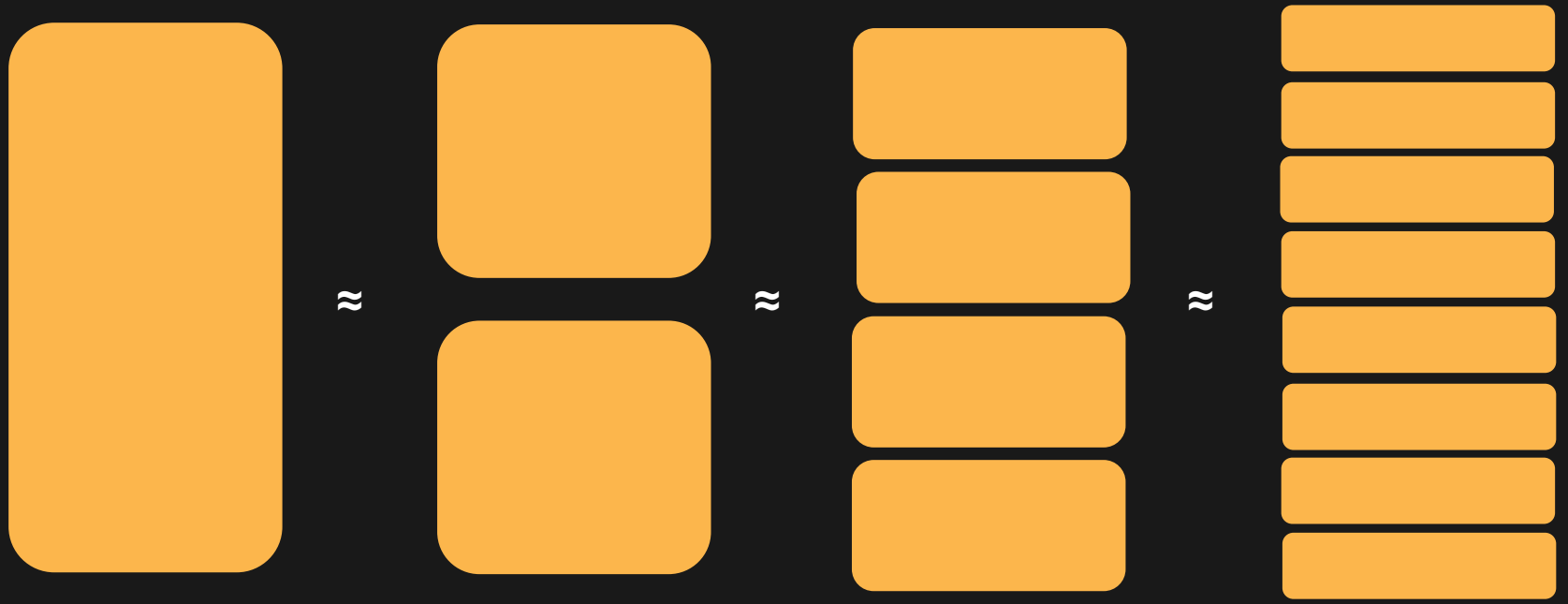
インスタンスファミリーの選択

システムのワークロードに合わせて最適なインスタンスファミリーを選択する



EC2インスタンスサイズ

C4インスタンスを例としたインスタンスサイズの成り立ち



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

物理サーバーあたりのインスタンス数について知りたい場合は「Dedicated Hosts」の情報を辿ることで確認が可能

EC2仮想サーバの物理リソース配分

- 各インスタンスに割り当てられたリソースは、オーバーコミットしておらず、インスタンスの所有者が占有して利用できる*
  - 全vCPUについてインスタンスのオーナーが占有利用
  - 割り当てられたメモリは、そのインスタンスで占有利用
  - ネットワークリソースは同居する他のインスタンスの影響を受けないよう分割されている

＊“T” インスタンスファミリーを除く

vCPU(Virtual CPU)とは

- vCPU数はハイパースレッド論理コアの数を指す*
- vCPU数を2で割った数が物理CPUコア数
- LinuxではCPU番号は物理CPUコア→論理CPUコアの順にオーダー
- Windowsでは物理CPUコア,論理CPUコア,物理,論理・・・の順にインターリーブされている

- Amazon EC2およびRDS DBインスタンスタイプ別の仮装コア数は下記のURLに記載:

  https://aws.amazon.com/ec2/virtualcores/

* “T” インスタンスファミリーを除く

ハイパースレッド無効化の方法

- 主に浮動小数点演算が多いアプリケーションで有効
- ‘lscpu’コマンドを使用してCPUレイアウトを確認
- 後半の論理CPUをオフラインにする

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

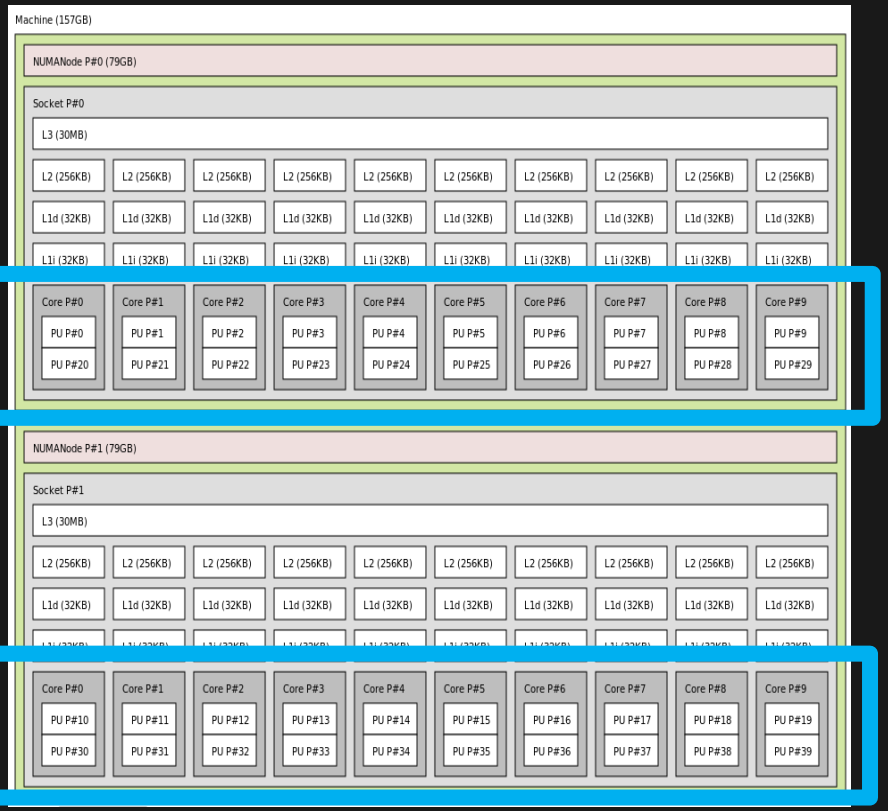
- もしくは全スレッドの半分だけ起動するようにGRUBのカーネルパラメータにセットすることでも調整可

  `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
```

vCPUの物理マッピングのイメージ

ハイパースレッド無効化前(Linux)

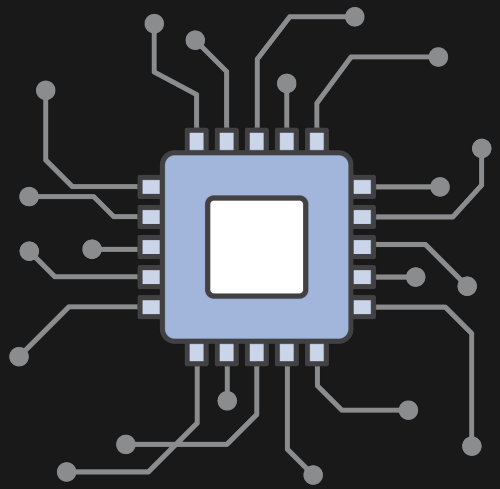


ハイパースレッド無効化後(Linux)

P-state と C-state の制御

- 対象インスタンスはc4.8xlarge, d2.8xlarge, m4.10xlarge, m4.16xlarge, p2.16xlarge, x1.16xlarge, x1.32xlarge
- 負荷が無いコアがより深いアイドル状態に遷移することにより、非アイドルコアはクロックアップして動作が可能(Turbo boost)
- ただし、より深いアイドル状態では終了に時間がかかり、レイテンシに敏感なワークロードには不適切な場合も考えられる
- C-stateの制御はGRUBのカーネルパラメータに下記を追加

  “intel_idle.max_cstate=1” to grub



Tips: AVX2のP-stateコントロール

- アプリケーションが全てのコアでAVX2を頻繁に使用すると、プロセッサはより多くの電力を消費しようとしてしまう可能性がある
- これによりプロセッサはクロック周波数を低くする動作をする
- CPUクロックを頻繁に変更するとアプリケーションが遅くなることがある
- 下記のコマンドによりターボブーストによるCPUクロックの変更を抑止可能

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

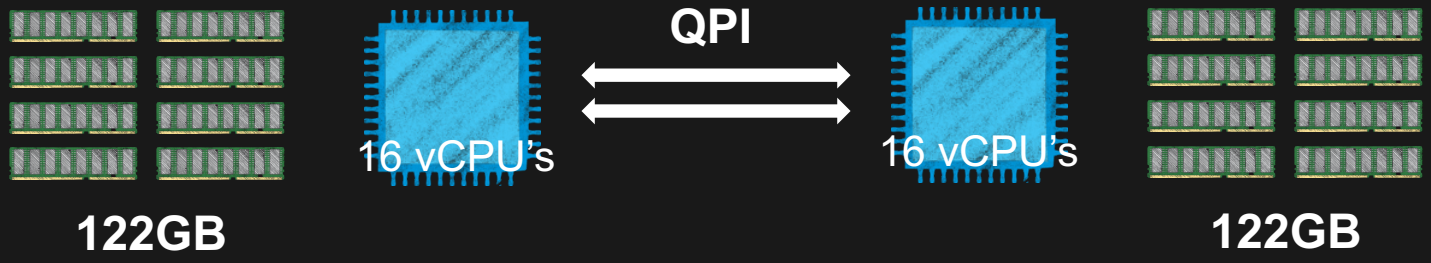
詳細は下記のドキュメントを参照:
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/processor_state_control.html

NUMA

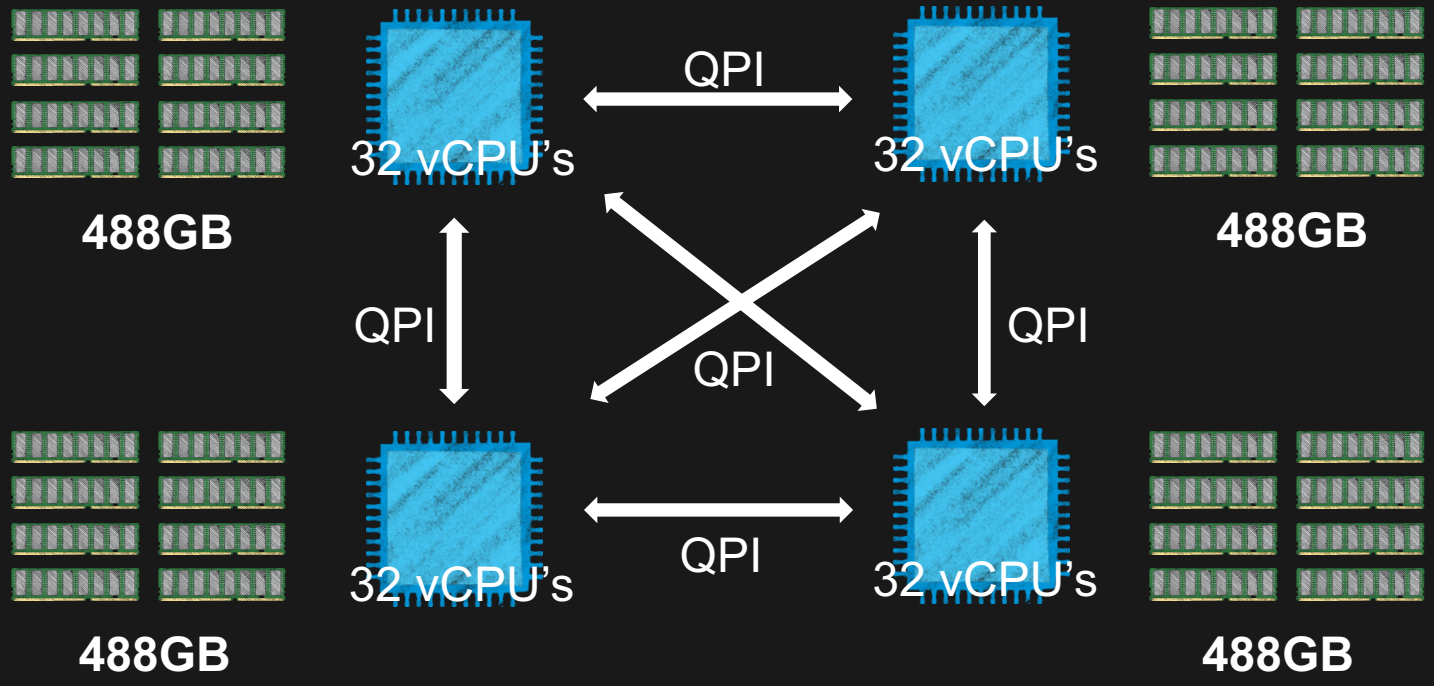
- Non-uniform memory accessの略
- マルチCPUシステムの各プロセッサには、内蔵したメモリコントローラにより直接アクセスできるローカルメモリが存在、プロセッサ間は高速インターコネクトで接続されている
- 各プロセッサは他のCPUからメモリにアクセスすることも可能だが、ローカルメモリアクセスはリモートメモリアクセスよりもはるかに高速
- パフォーマンスはCPUソケットの数とそれらの接続方法に関連している - Intel QuickPath Interconnect (QPI)

→ 次ページより詳細説明

r3.8xlarge



x1.32xlarge


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

Tips: カーネルのNUMAバランシングサポートについて

- プロセスのスレッドが同じNUMAノード上のメモリにアクセスしているときに、アプリケーションは最高のパフォーマンスを発揮する
- NUMAバランシングは、タスクをアクセスしているメモリに近づける
- Linuxカーネルのバージョンが3.8以降の場合、自動でNUMAバランシングが有効となる
- WindowsでのNUMAサポートはWindows Server 2003 Enterprise ならびに Data Center で最初に行われた
- アプリケーションが単一のソケットに収まるよりも多くのメモリを使用している場合、またはソケット間を移動するスレッドがある場合はNUMAページングを減らす為 “numa = off”(kernelパラメータ) を設定する
- 単一ソケット内で完結するプロセスを起動する場合
  - 下記のnumactlコマンドで特定コア-ノードにプロセスの動作を制限
    - 例: `numactl --cpunodebind=0 --membind=0 ./myapp.run`

アジェンダ

- EC2インスタンスリソースよる性能要素
- I/O関連のパフォーマンスオプション
- パフォーマンスクレジット
- その他のパフォーマンス要素
- まとめ

ストレージ関連

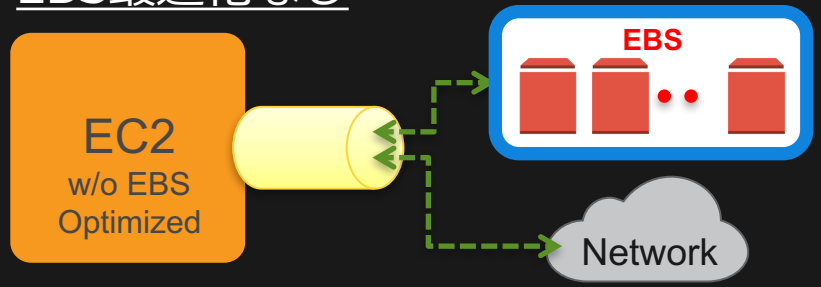
EBSパフォーマンス影響要素

- インスタンスサイズがEBSのスループットに影響する
- ボリュームサイズやボリュームのタイプによっても性能変化
- EBSのパフォーマンスが重要な場合はEBS最適化を使用する

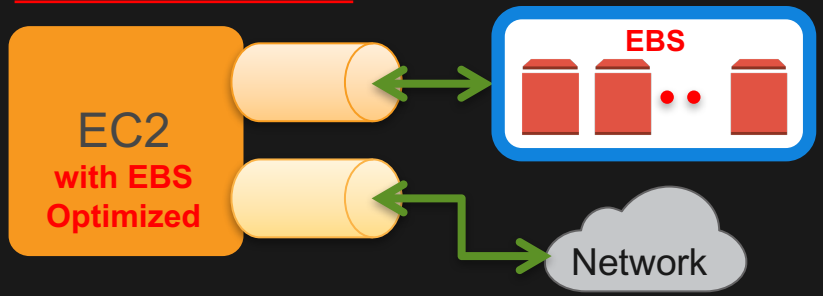
Instance type	EBS-optimized by default	Max. bandwidth (MiB/s)*	Max. IOPS (16 KiB I/O size)*	Throughput (Mbps)**
c3.xlarge		62.5	4,000	500
c3.2xlarge		125	8,000	1,000
c3.4xlarge		250	16,000	2,000
c4.large	Yes	62.5	4,000	500
c4.xlarge	Yes	93.75	6,000	750
c4.4xlarge	Yes	250	16,000	2,000
c4.8xlarge	Yes	500	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

EBS最適化(EBS-Optimized)

EBS最適化なし



EBS最適化あり



EBS最適化を有効にすることで独立した帯域を確保しI/O性能の安定化に繋がる

大きいインスタンスタイプほど使える帯域が広い(*)

特別な意図が無ければ基本的に有効に設定することが望ましい

最近のインスタンスタイプ(m4,c4,d2)ではデフォルトでオンになっている

(*)インスタンス毎の帯域幅はこちらを参照ください
http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/EBSOptimized.html

<参考>Amazon EBS – 各ボリュームタイプの特徴

コストを考慮しつつ、必要な性能に合せたボリュームタイプを選択する

	Solid State Drive (SSD)		Hard Disk Drive (HDD)	
ボリュームタイプ	**プロビジョンド IOPS SSD (io1)**	**汎用SSD (gp2)**	**スループット最適化HDD (st1)**	**コールドHDD (sc1)**
ユースケース	I/O性能に依存するNoSQLデータベースやリRDB	起動ボリューム、低レイテンシを要求するアプリ、開発・テスト環境	ビッグデータ、DWH、ログデータ処理	アクセスする頻度が低いデータ
ボリュームサイズ	4 GB – 16 TB	1 GB – 16 TB	500 GB – 16 TB	500 GB – 16 TB
ボリューム毎の最大IOPS	20,000 (16 KB I/O size)	10,000 (16 KB I/O size)	500 (1 MB I/O size)	250 (1 MB I/O size)
インスタンス毎の最大IOPS(複数ボリューム)	48,000	48,000	48,000	48,000
ボリューム毎の最大スループット	320 MB/s	160 MB/s	500 MB/s	250 MB/s
月額料金 (東京リージョン)	$0.142/GB + $0.074/設定IOPS値	$0.12/GB	$0.054/GB	$0.03/GB
性能指標	IOPS	IOPS	MB/s	MB/s

st1,sc1ボリュームにおけるパラメータチューニング(1)

- 高スループットで読み込みが主体となるワークロードにおいては、性能を最大限引き出すために先読み(Read Ahead)のサイズを1MBに設定することを推奨

1. 現状の設定を確認する
   $ sudo blockdev --report /dev/(device)

2. Read aheadの値を変更する
   $ sudo blockdev --setra 2048 /dev/(device)

3. 設定変更結果を確認する
   $ sudo blockdev --report /dev/(device)

```
[ec2-user@ip-172-31-31-63 ~]$ sudo blockdev --report /dev/xvde
RO    RA   SSZ   BSZ   StartSec            Size   Device
rw  2048   512   512          0   10995116277760   /dev/xvde
```

詳細情報：http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/EBSPerformance.html

st1,sc1ボリュームにおけるパラメータチューニング(2)

- Linux Kernelのバージョン4.2以上を利用している場合は、先の設定に加えて*xen_blkfront.max*の値を*256*に設定することを推奨
- この値はカーネルモジュールパラメータとして指定を行う。Amazon Linuxの場合は下記の手順で設定変更が可能

1. /boot/grub/menu.lstをviで開く
   $ sudo vi /boot/grub/menu.lst

2. kernel行を以下の通り追記してOSを再起動
   変更前) kernel /boot/vmlinuz-4.4.5-15.26.amzn1.x86_64 root=LABEL=/ console=ttyS0
   変更後) kernel /boot/vmlinuz-4.4.5-15.26.amzn1.x86_64 root=LABEL=/ console=ttyS0 **xen_blkfront.max=256**

詳細情報：http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/EBSPerformance.html

インスタンスストア

ローカルディスクに大量の一時ファイル(スクラッチファイル)を読み書きする事が多いアプリケーションを実行する場合、演算で使用するスクラッチディスクとしてEC2インスタンス内蔵ディスクであるインスタンスストアの利用が効果的。

<ポイント>
・インスタンス内蔵ディスクなので、ネットワーク性能の影響を受けず高速アクセスが可能
・費用がインスタンスの料金に含まれている

※インスタンスストアはインスタンスの停止/起動の都度、内容が消去される為、扱いに注意

インスタンスタイプ	CPU	物理コア数(vCPU数)	メモリ[GiB]	インスタンスストア	ネットワーク性能
I3 High I/Oインスタンス	Xeon E5-2686v4	～32 (64)	～488	～1.9TB NVMe SSD×8	～20Gbps
R3 メモリ最適化	Xeon E5-2670v2	～16 (32)	～244	～320GB SSD×2	～10Gbps
X1 メモリ最適化	Xeon E7-8880v3	～64 (128)	～1,952	～1,920GB SSD×2	～20Gbps

上記以外にもインスタンスストレージを搭載したインスタンスは存在。下記EC2インスタンスの情報を参照。
https://aws.amazon.com/jp/ec2/instance-types/

ネットワーク関連

インスタンスのサイズとネットワーク性能の関係

インスタンスのネットワーク性能は、例えば同じc4インスタンスでも、インスタンスのサイズによって性能は異なる。インスタンスのサイズに比例して、インスタンスあたりのネットワーク性能も向上する為、並列処理などインスタンス間の通信性能が影響する場合、CPU/メモリ量に関わらずインスタンスサイズの考慮が必要。

C4インスタンスでのネットワークパフォーマンスの違い

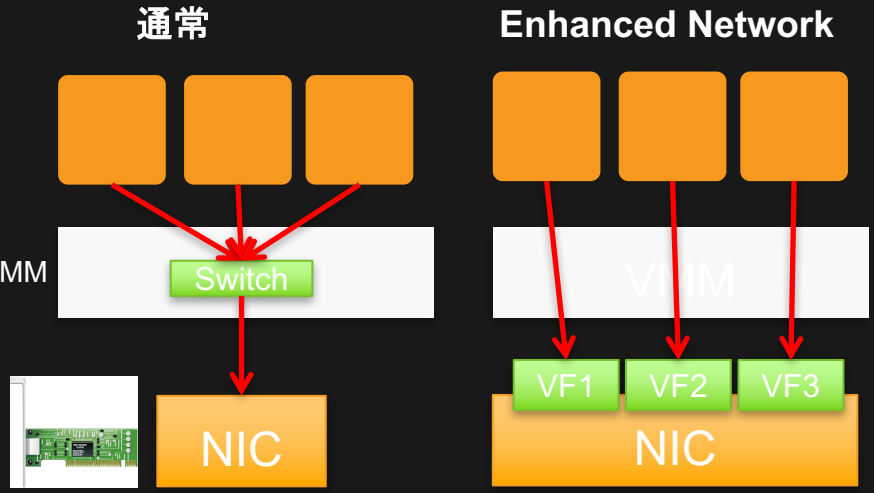
インスタンスタイプ	vCPU	メモリ [GiB]	ネットワーキング パフォーマンス
c4.large	2	3.75	中
c4.xlarge	4	7.5	高
c4.2xlarge	6	15	高
c4.4xlarge	16	30	高
c4.8xlarge	36	60	10ギガビット

各インスタンスのネットワーク性能については下記の中のインスタンスタイプマトリックスを参照
https://aws.amazon.com/jp/ec2/instance-types/

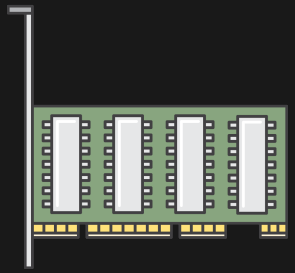
拡張ネットワーキング

今現在、拡張ネットワーキング機能には下記2タイプが存在

- Intel 82599VF (ixgbevf)
  - 最大10Gbpsのネットワーク速度をサポート
  - パケット毎秒(PPS)が非常に大きく、ネットワークレイテンシが低くなるオプション
  - SR-IOVに対応



- Elastic Network Adapter (ENA)
  - 最大20Gbps(今現在)のネットワーク速度
  - マルチキューデバイスインタフェース
  - ハードウェアによるIPv4ヘッダーならびに一部のTCP/UDPチェックサム生成をサポート



拡張ネットワーキングの前提条件

- 対応インスタンスタイプ:
  - ixgbevf:　C3, C4, D2, I2, M4 (except m4.16xlarge), R3
  - ENA:　　m4.16xlarge, P2, X1, R4, I3, F1
- OSに要件を満たすixgbevf または ENA ドライバの導入
  - Ixgbevf:　ドライバ バージョン 2.14.2 以降
  - ENA:　　ドライバは全リリース対応
- AMIタグまたはインスタンス属性に拡張ネットワーキング有効化オプション付与

拡張ネットワーキング有効化確認 (インスタンス属性)

82599 Enhanced Networking

```
% aws ec2 describe-instance-attribute --instance-id instance_id ¥
  --attribute sriovNetSupport
{
    "InstanceId": "instance_id",
     "SriovNetSupport": {
        "Value": "simple"
    }
}
```

82599VF 有効!

ENA Enhanced Networking

```
% aws ec2 describe-instances --instance-id instance_id ¥
  --query "Reservations[].Instances[].EnaSupport"
[
    true
]
```

ENA 有効!

拡張ネットワーキング有効化確認(AMI属性)

- 82599 Enhanced Networking:

```
% aws ec2 describe-image-attribute --image-id ami_id ¥
  --attribute sriovNetSupport
```

- ENA Enhanced Networking:

```
% aws ec2 describe-image-attribute --image-id ami_id ¥
  --attribute enaSupport
```

OS上からの拡張ネットワーキング有効化確認 (ixgbevf)

NGな例

```
[ec2-user ~]$ ethtool -i eth0
driver: vif
version:
firmware-version:
bus-info: vif-0
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
```

OKな例

```
[ec2-user ~]$ ethtool -i eth0
driver: ixgbevf
version: 2.14.2
firmware-version: N/A
bus-info: 0000:00:03.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: no
supports-register-dump: yes
supports-priv-flags: no
```

OS上からの拡張ネットワーキング有効化確認 (ENA)

NGな例

```
[ec2-user ~]$ ethtool -i eth0
driver: vif
version:
firmware-version:
bus-info: vif-0
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
```

OKな例

```
[ec2-user ~]$ ethtool -i eth0
driver: ena
version: 0.6.6
firmware-version:
bus-info: 0000:00:03.0
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
```

拡張ネットワーキング有効化手順

1. インスタンスを起動する
2. ENAカーネルドライバをビルドしインストールする
3. 新しいAMIを作成し変更するかインスタンスを変更する

```
# aws ec2 modify-instance-attribute ¥
  --instance-id instance_id --ena-support
```

または

```
# aws ec2 register-image --ena-support ...
```

4. 再起動またはAMIからインスタンスを作成して利用する

詳細は下記ドキュメントを参照:

<Linux>

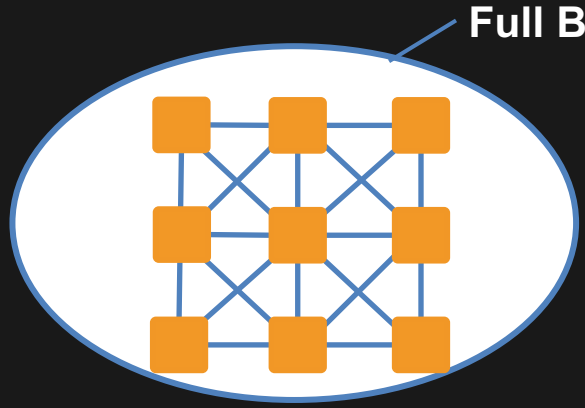
http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/enhanced-networking.html#enabling_enhanced_networking

<Windows>

http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/WindowsGuide/enhanced-networking.html

プレイスメントグループ

- インスタンス間通信を最適化するオプション
- 広帯域 (最大10 or 20Gbps Full Bisection)
- 低レイテンシ
- 高PPS (packets per seconds)



対応インスタンス

・C4、C3、cc2、X1、R3、cr1、I2、D2、hs1、hi1
P2、G2、cg1、M4、R4、I3、F1

使い方

所属させたいプレイスメントグループ名を、
インスタンス作成時に「配置グループ」の項目で新規作成
もしくは、既存のプレイスメントグループから名前を選択

※詳細ガイド→ http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/placement-groups.html

ネットワークパフォーマンス-その他ポイント

- 20 Gigabit & 10 Gigabit
  - 片方向通信の速度、双方向ではこの倍の帯域を持つ(full duplex通信)
- 高速、中速、低速 – インスタンスサイズとEBS最適化のスループット表記
  - 全てが等しく作られている訳ではない – 性能が重要な場合はiperfでテストを行うこと
- インスタンス同士通信において一貫した帯域幅が必要な場合はプレイスメントグループを使用する、またプライベートIPで通信すること
- EC2外への通信(S3など他のサービスetc.)は5Gb/sの帯域に制限される

アジェンダ

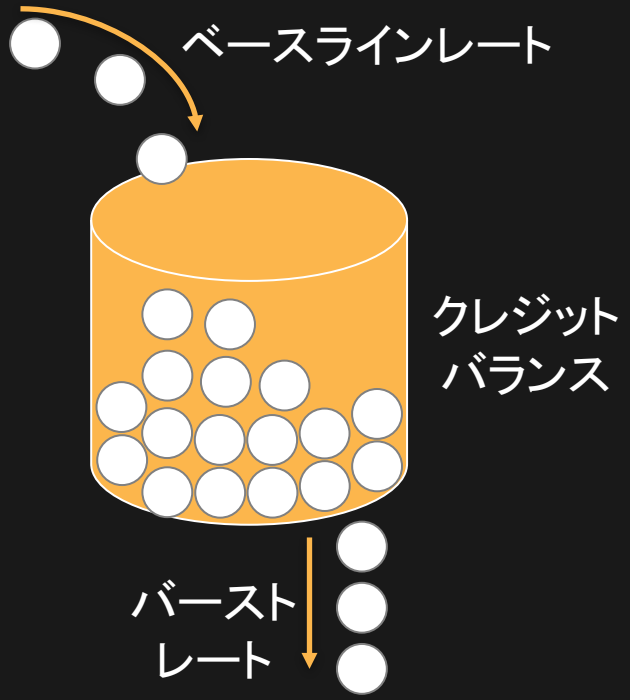
- EC2インスタンスリソースよる性能要素
- I/O関連のパフォーマンスオプション
- パフォーマンスクレジット
- その他のパフォーマンス要素
- まとめ

T2インスタンス

- T2インスタンスはEC2インスタンスの最低コストとなる$0.0065/時間～
- ユースケース→汎用処理、Webサーバー、開発環境、小さなデータベース
- **CPUクレジットによりCPU性能がバーストする**

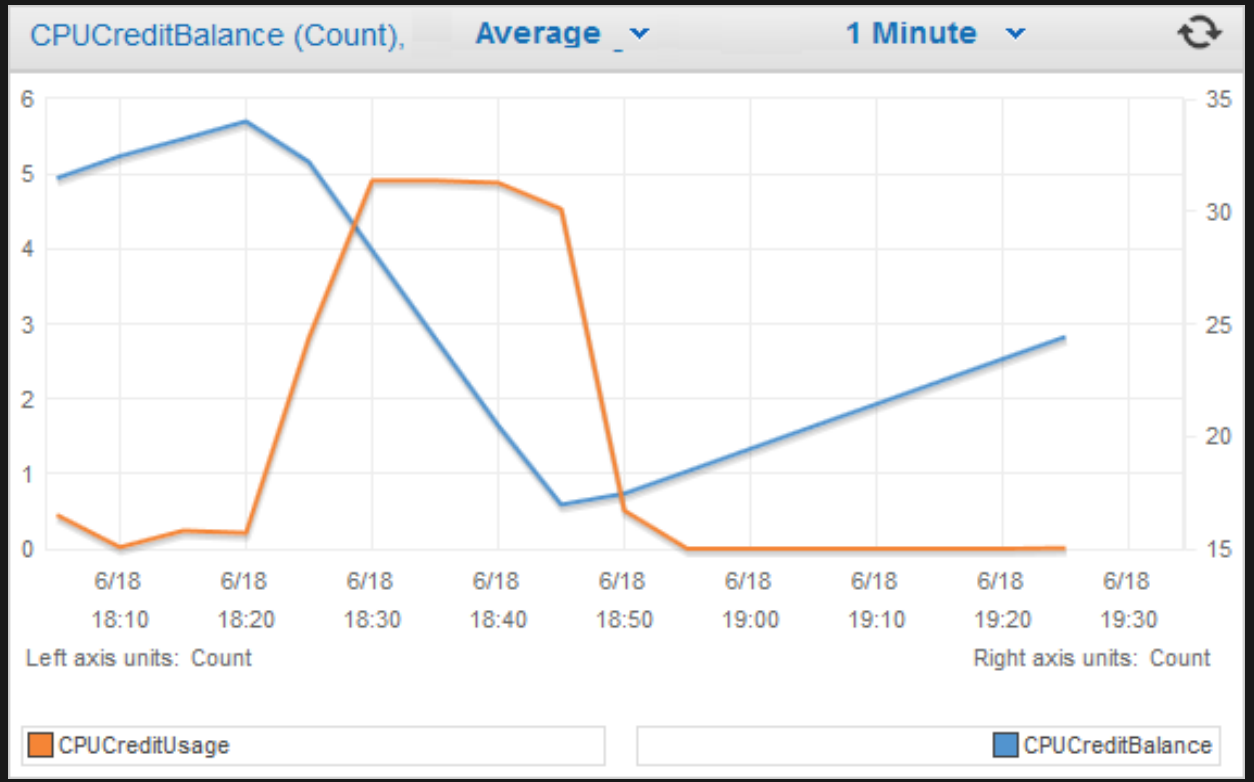
Model	vCPU	Baseline	CPU Credits / Hour	Memory (GiB)	Storage
t2.nano	1	5%	3	.5	EBS のみ
t2.micro	1	10%	6	1	EBS のみ
t2.small	1	20%	12	2	EBS のみ
t2.medium	2	40%**	24	4	EBS のみ
t2.large	2	60%**	36	8	EBS のみ

クレジットの動作



- CPUクレジットはCPUコアの最大パフォーマンス(バースト性能)を1分間提供する
- インスタンスは一定の割合でCPUクレジットを獲得する
- インスタンスがバースト時にクレジットを消費する
- クレジットは24時間で失効する(リーク)

<参考>CPUクレジットのモニタリング結果



その他のクレジット(EBSボリューム性能)

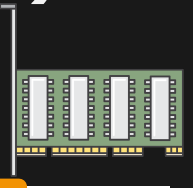
SSDタイプのgp2ボリュームならびにHDDタイプのst1およびsc1ボリュームには、それぞれ下記のクレジットが存在

- SSDタイプ gp2　　：ボリュームI/Oクレジット
- HDDタイプ st1,sc1　　：ボリュームスループットクレジット

これらのバーストクレジットを消費すると、ベースラインにスロットリングされる。
EBSのバーストクレジットの残高はCloudWatchで確認可能。(BurstBalance値)

詳細は下記EBSのBlackBelt資料を参照：
https://www.slideshare.net/AmazonWebServicesJapan/aws-black-belt-online-seminar-amazon-elastic-block-store-ebs

その他のクレジット(R4インスタンスの通信帯域)



- r4.8xlargeは10Gbpsの帯域
  r4.16xlargeは20Gbpsの帯域
- 上記より小さいインスタンス
  - 最大10 Gbps のベースライン性能
  - ベースラインを下回った場合にクレジットを獲得

Model	vCPU	Mem (GiB)	Networking Performance	SSD Storage (GB)
r4.large	2	15.25	Up to 10 Gigabit	EBS-Only
r4.xlarge	4	30.5	Up to 10 Gigabit	EBS-Only
r4.2xlarge	8	61	Up to 10 Gigabit	EBS-Only
r4.4xlarge	16	122	Up to 10 Gigabit	EBS-Only
r4.8xlarge	32	244	10 Gigabit	EBS-Only
r4.16xlarge	64	488	20 Gigabit	EBS-Only

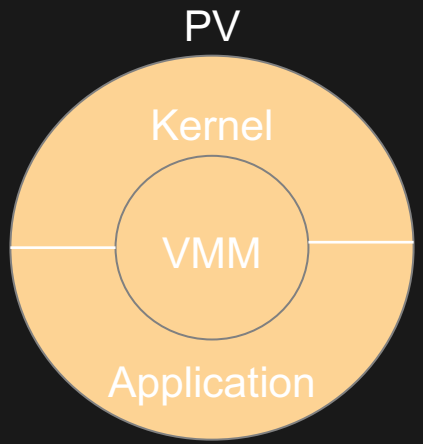
詳細は下記ドキュメントを参照：
http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/memory-optimized-instances.html

アジェンダ

- EC2インスタンスリソースよる性能要素
- I/O関連のパフォーマンスオプション
- パフォーマンスクレジット
- その他のパフォーマンス要素
- まとめ

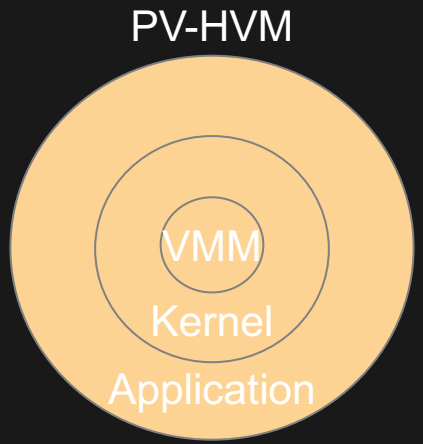
X86 CPUの仮想化機能: Intel VT-x が出る以前

- 特権命令でのバイナリ変換
- 準仮想化(PV)
  - PVはVMMを通過する為、レイテンシが加わる
  - システムコールバインドされているアプリケーションが最も影響を受ける



X86 CPUの仮想化機能: Intel VT-x 機能が出て以降

- ハードウェアアシスト仮想化 (HVM)
- PV-HVMでは、エミュレートが遅い操作に対して、PVドライバを便宜的に使用する:
  - e.g., ネットワークやブロック I/O



Tips: HVM AMIを使用する

同種のOSでも仮想化タイプが異なるAMIがリリースされている事があるので確認し選択

Amazon Linux AMI IDs

Amazon Linux AMI 2015.09 was released on 2015-09-22.

Region	HVM (SSD) EBS-Backed 64-bit	HVM Instance Store 64-bit	PV EBS-Backed 64-bit	PV Instance Store 64-bit	HVM (Graphics) EBS-Backed 64-bit
US East N. Virginia	ami-e3106686	ami-65116700	ami-cf1066aa	ami-971066f2	AWS Marketplace
US West Oregon	ami-9ff7e8af	ami-bbf7e88b	ami-81f7e8b1	ami-bdf7e88d	AWS Marketplace
US West N. California	ami-cd3aff89	ami-c33aff87	ami-d53aff91	ami-c93aff8d	AWS Marketplace
EU Ireland	ami-69b9941e	ami-7db9940a	ami-a3be93d4	ami-8fbe93f8	AWS Marketplace

オペレーティングシステムの性能への影響について

ここで下記のベンチマークを行い、実際に性能比較を行う

- メモリインテンシブなWebアプリケーション
  - 多くのスレッドを生成
  - メモリを頻繁に割り当て/解放

  (類似の特性を持つベンチマークツール”ebizzy“を使用)
- RHEL6とRHEL7のパフォーマンス比較
- 出力で “system” 時間を確認
- “perf”コマンドにて性能分析

RHEL6の結果

```
[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
```

RHEL7の結果

```
[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
```

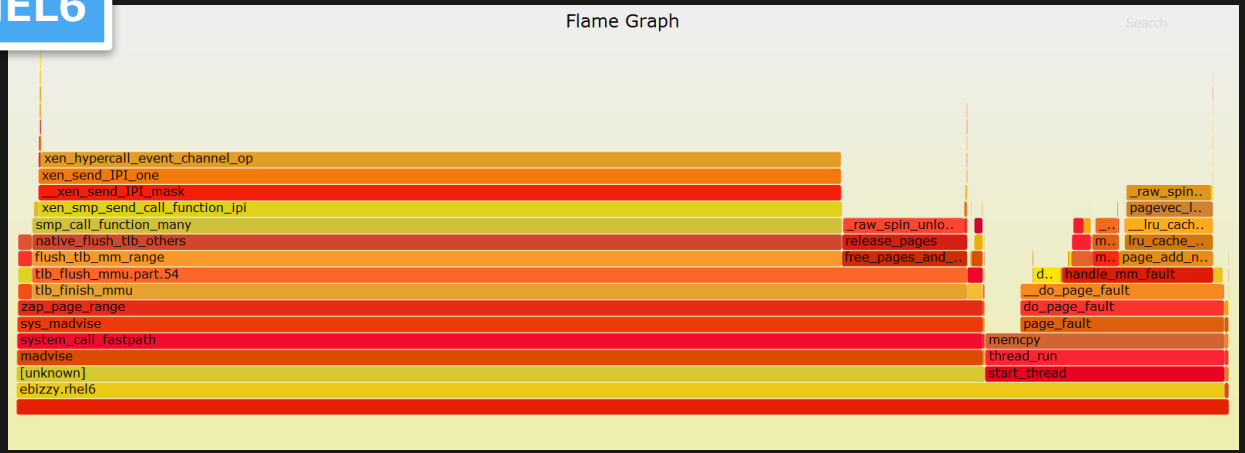
RHEL6での12,409 records/s
から大幅にスループット向上

RHEL6での1,418,204から
大幅に減少

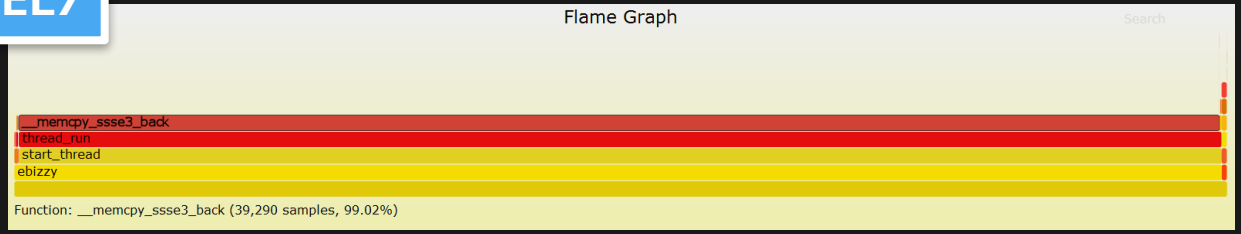
Flame Graph の出力

www.brendangregg.com/flamegraphs.html

RHEL6



RHEL7



RHEL7では新しいカーネルバージョンにより、特にxen関連のコールやflushなどのシステムコールが無くなり、オーバーヘッドが減少していることが解る

Tips:オペレーティングシステムの選択

- RHEL6のLinuxカーネルバージョンは2.6.32。これは2009年のものであり、特に仮想化環境で利用する場合は性能に多少難あり

- なるべく新しいカーネルを搭載したOSを選択する(3.10以降)
  - Amazon Linux 13.09以降
  - Ubuntu 14.04 以降
  - RHEL/CentOS 7 以降
  - etc.

時刻情報の問い合わせ処理について

- インスタンス上からの時刻情報の問い合わせ処理はとても複雑
- gettimeofday(), clock_gettime(), QueryPerformanceCounter()
- 現行世代のインスタンスでは、TSCをクロックソースとして使用可

- TSCとは
  - CPUカウンタ
  - vDSO(virtual Dynamic Shared Object)をサポートしユーザ空間で処理可能

- Xen pvclock はvDSOをサポートしていない

タイムインテンシブなプログラムのベンチマーク

```
#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クロックソースを使用した場合

```
[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クロックソースを使用した場合

```
[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
```

タイムインテンシブなプログラムのベンチマーク(結果比較)

Xenクロックソースを使用

```
[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クロックソースを使用

```
[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

Tips: TSCクロックソースを使用する方法

現状確認:

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

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

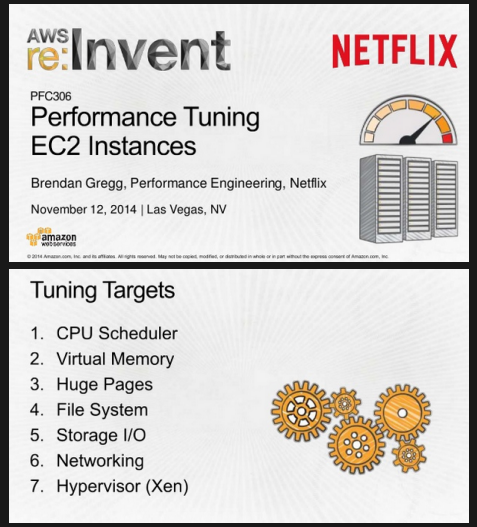
変更:

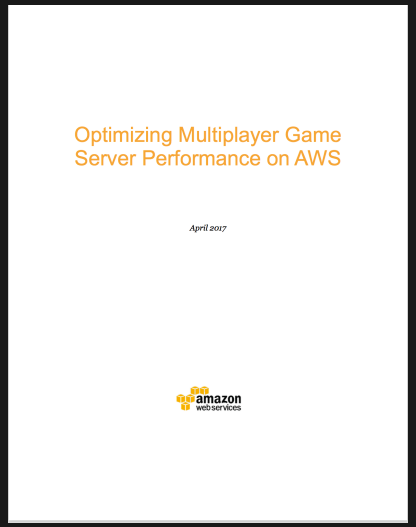
```
# echo tsc > /sys/devices/system/cl*/cl*/current_clocksource
```

その他チューニングの際にヒントとなる情報

Netflixのチューニング事例資料
http://www.slideshare.net/brendangregg/performance-tuning-ec2-instances#35

Optimizing Multiplayer Performance on AWS
https://d0.awsstatic.com/whitepapers/optimizing-multiplayer-game-server-performance-on-aws.pdf





※チューニングはやみくもに行うものではなく、アプリケーション実行中にリソースモニタリングツール(OSのvmstatやsarコマンドもしくはCloudWatchなど)でCPU/メモリ/通信の状態を計測し、ボトルネックとなっている部分をチューニングしていくことが基本。

アジェンダ

- EC2インスタンスリソースよる性能要素
- I/O関連のパフォーマンスオプション
- パフォーマンスクレジット
- その他のパフォーマンス要素
- まとめ

まとめ: EC2インスタンスを最大限に活用する

今回紹介した内容のサマリー

- 性能要素を理解した最適なインスタンス選択
- ハイパースレッドやC-state と P-stateなどCPU機能の理解
- 物理的なメモリ配置を意識したNUMAバランシングの制御
- ストレージ、ネットワーク性能オプションの理解
- T2のCPUグレジットをはじめとした性能クレジット
- AMIによる仮想化タイプの違い
- OSバージョンやオプションで変化する性能
- 使用するアプリケーションをプロファイルしてチューニングする


AWS
SUMMIT
amazon
web services

本セッションのFeedbackをお願いします

受付でお配りしたアンケートに本セッションの満足度やご感想などをご記入ください
アンケートをご提出いただきました方には、もれなく素敵なAWSオリジナルグッズを
プレゼントさせていただきます

アンケートは受付、パミール3FのEXPO展示会場内にて回収させて頂きます