PCクラスター構築入門

樫村 博基

- **注意**
  - 本発表の目的は、「初めてPCクラスタを構築した」経験を共有することです。
  - 発表者は、計算機についてそんなに詳しくありません。

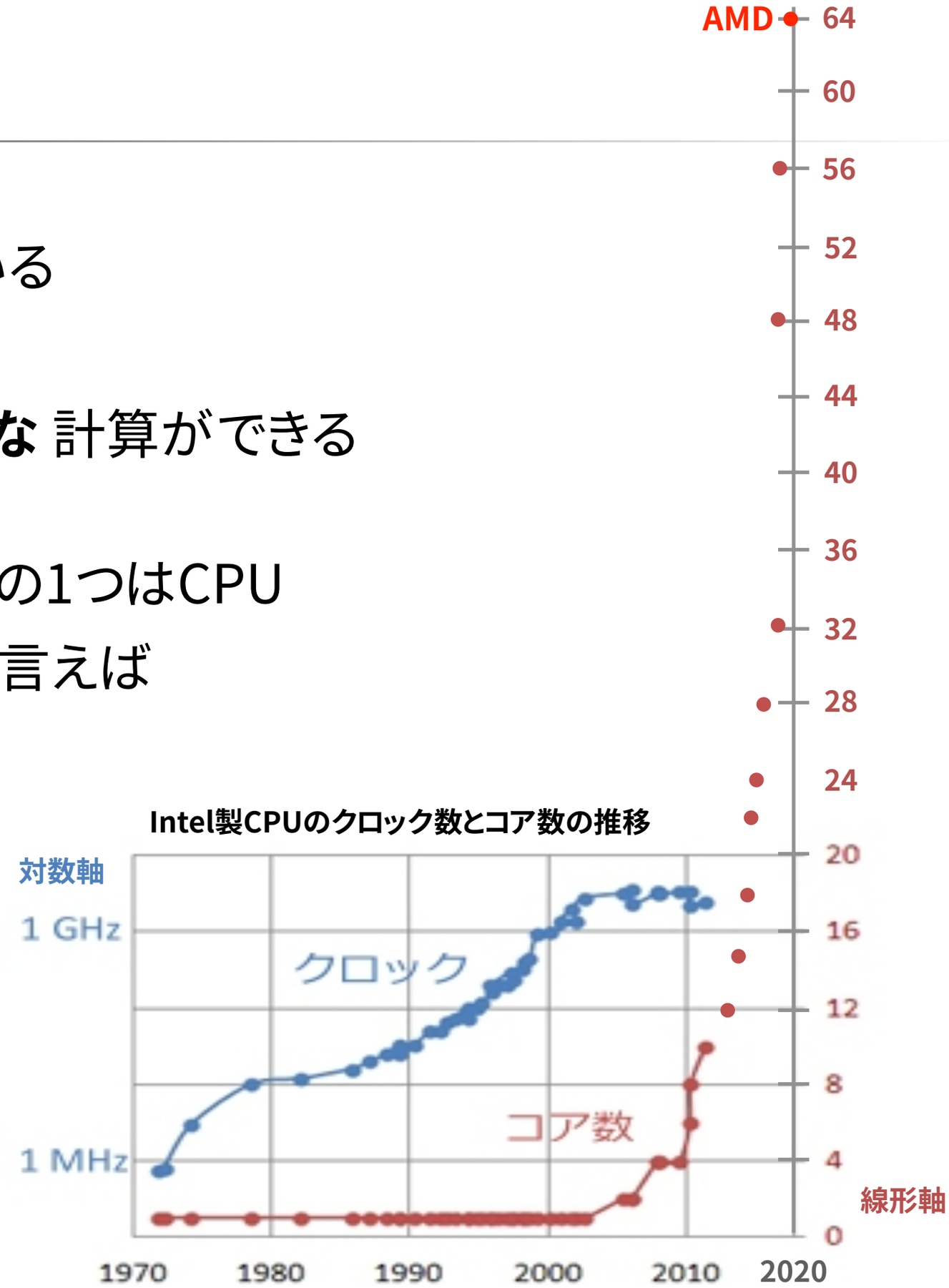
はじめに (1)

- 数値計算・シミュレーションは、
  さまざまな学問分野で利用されている
- 計算機の能力が高いほど
  - **より速く、より多くの、より大規模な** 計算ができる

- 計算機の能力を決める主要な部品の1つはCPU
  - CPUの性能は、ものすごくざっくり言えば

クロック数(1秒間に計算できる回数)
×
コア(演算装置)の数

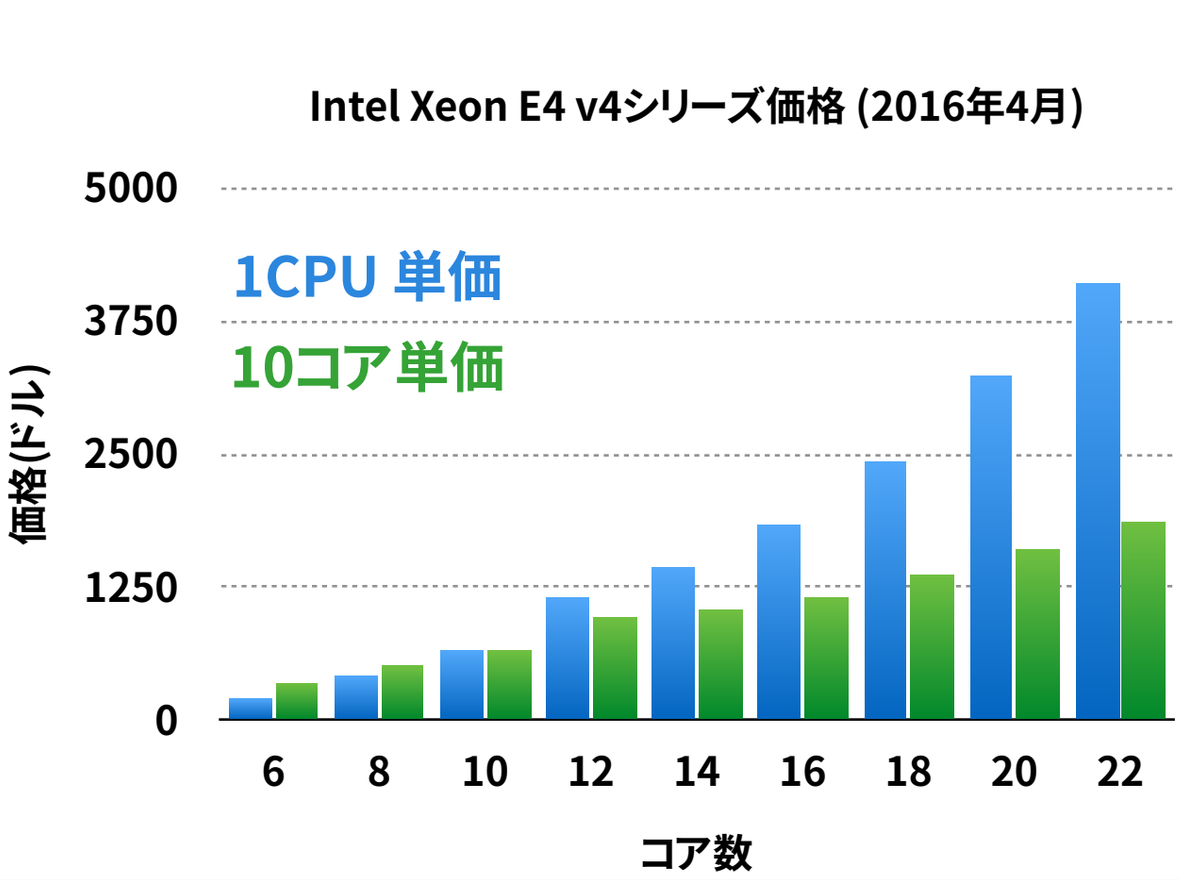
で決まる

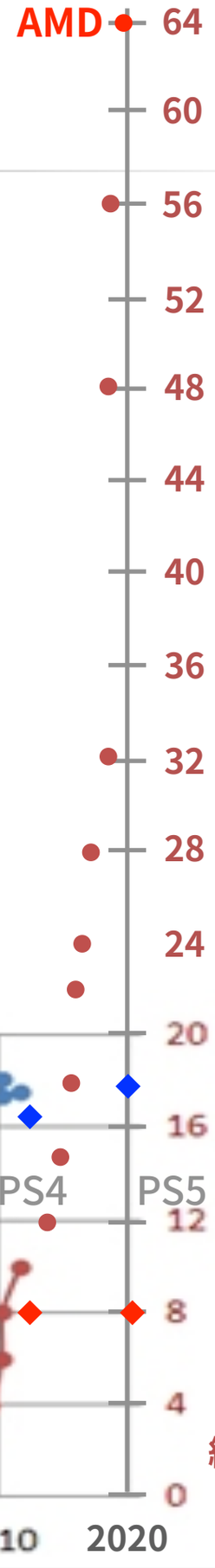


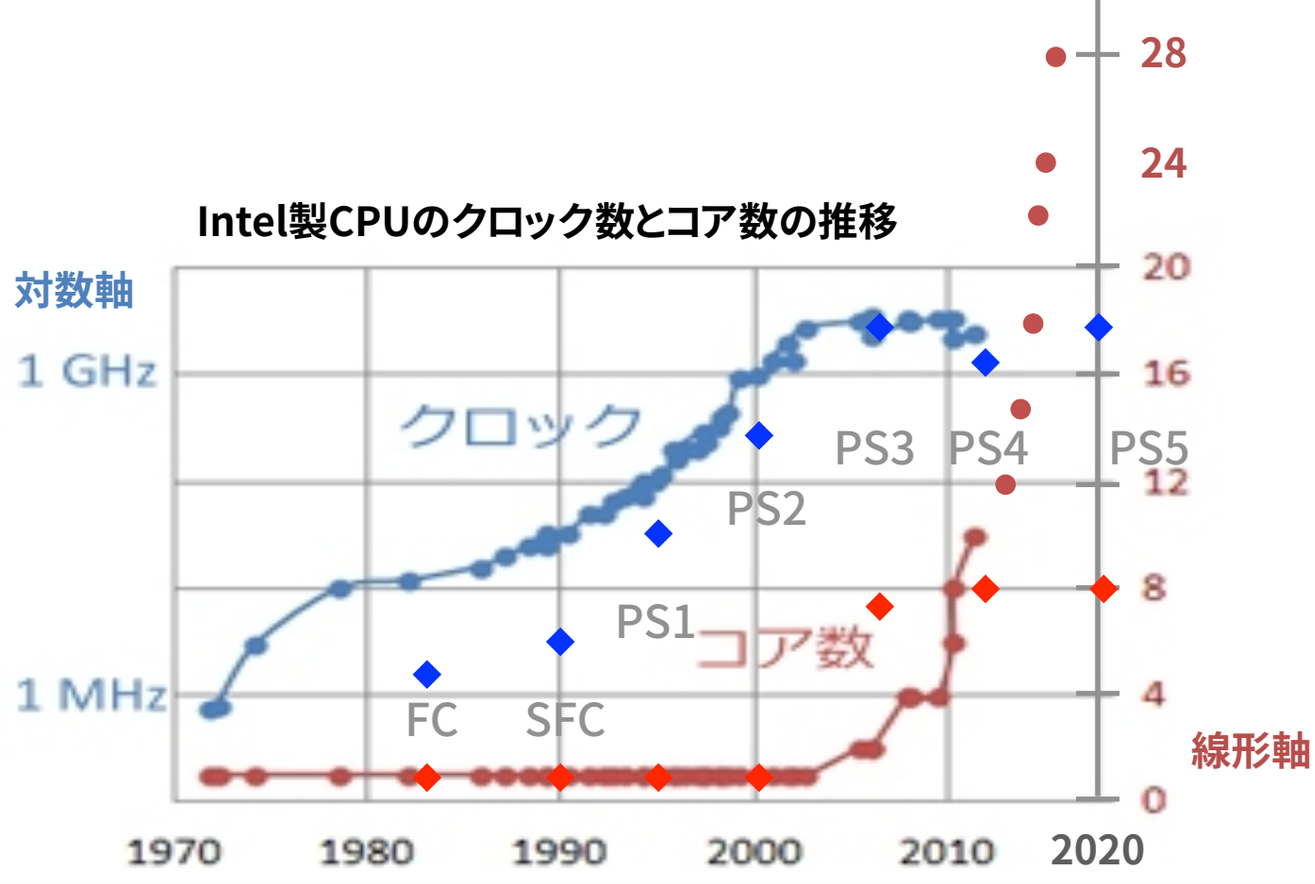
グラフ：https://www.rccm.co.jp/development/parallel/

はじめに (2)

- 2003年頃までは、**クロック数** は指数関数的に増加
  - プログラムは勝手に速くなる
- 以後、クロック数は頭打ち。かわりに **コア数** が増加
  - **並列計算** をしないと、計算機性能を活かせない
- コア数の多いCPUは価格も高い、コア単価も高い



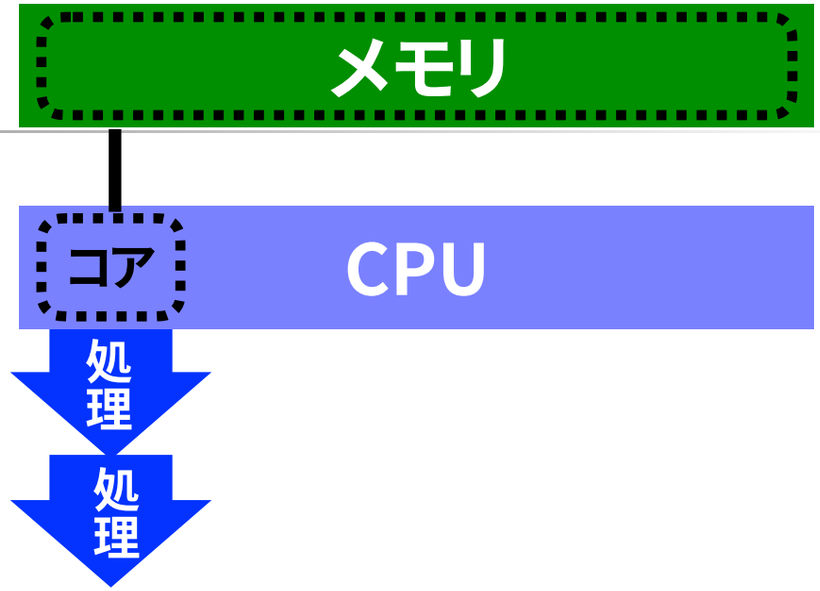




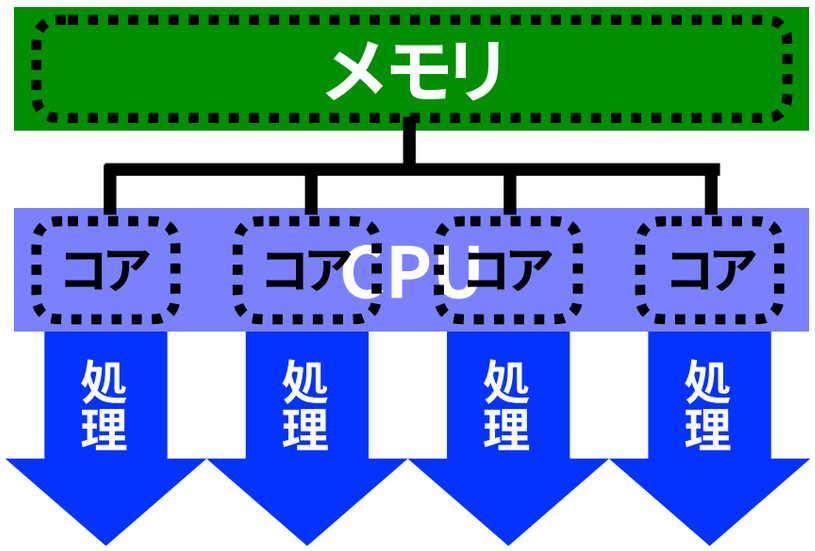
参考: https://www.4gamer.net/games/107/G010793/20160331095/　グラフ: https://www.rccm.co.jp/development/parallel/

逐次計算と並列計算

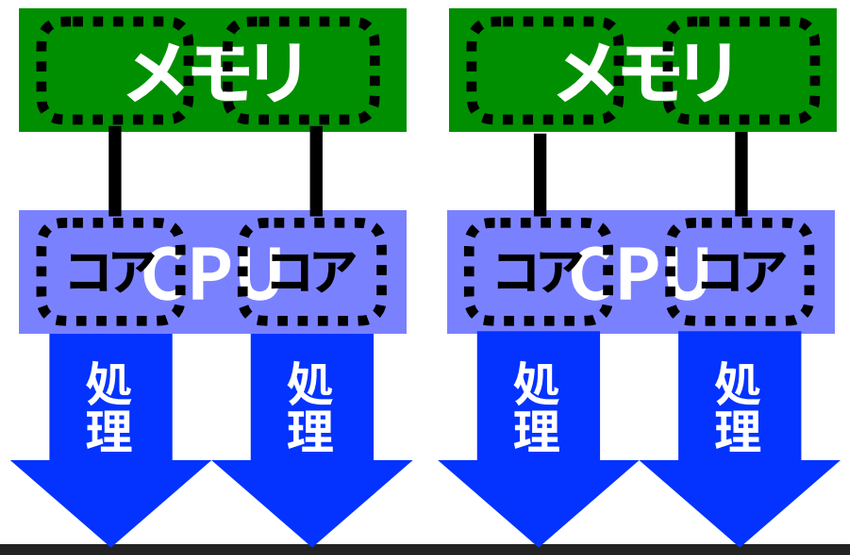
- **逐次計算**
  - 1つのCPUコアが順番に(切り替えながら)処理



- **メモリ共有型並列計算｜OpenMP**
  - 複数のコア(同じPC内)が並列に処理
  - 各処理は同じメモリ領域(変数)を参照できる
  - 最多並列数は、PC当たりのコア数

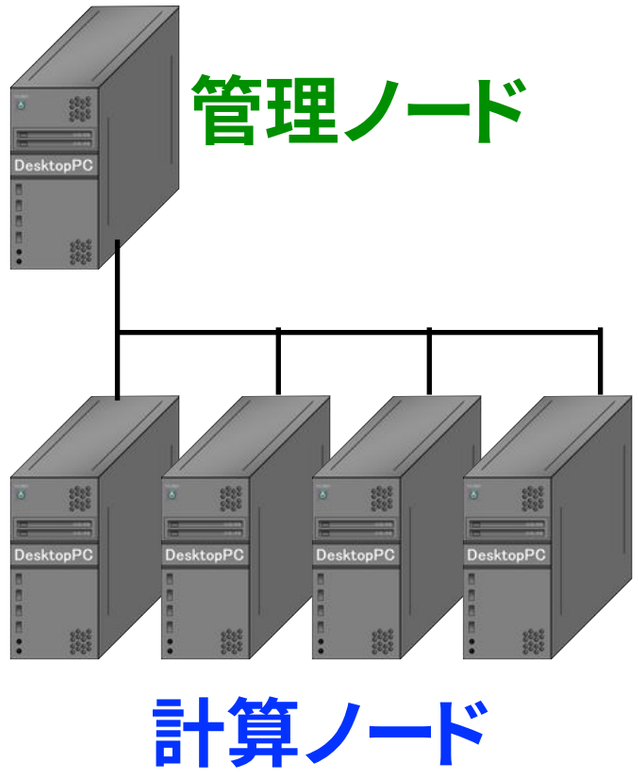


- **メモリ分散型並列計算｜MPI**
  - 複数のコア(別PCでも可)が並列に処理
  - メモリ領域は処理ごとに独立
  - 処理の前後で 通信が必要
  - マシンを増やせば多並列計算が可能



PCクラスター

- 複数のPCを繋いで、並列計算できるようにしたもの
  - **管理ノード**・・・並列処理を管理するPC
  - **計算ノード**・・・処理を行うPC
  - （メモリ領域を共有しているPC単位をノードと呼ぶ）



- ノード間の繋ぎ方 (代表的な通信速度)
  - Ethernet (1 Gbps, 10 Gbps)
  - Myrinet* (20 Gbps)
  - InfiniBand (~100 Gbps)
  - Omni-Path (~100 Gbps)

- 大型計算機も原理は同じ

*ミリネットと読む

個人的な背景と動機

- これまでに扱ってきた数値計算用マシン
  - 学生時代
    - 研究室(石岡先生)管理のワークステーション や MacPro
  - その後
    - 「京」や「富岳」などの大型計算機（スーパーコンピュータ）
  - 昨年
    - セット売りのPCクラスター
      - ✓ 並列計算環境は構築済み・・・**『OSやコンパイラは更新しないでください』**😨
- **環境構築手順を習得したい&テスト環境が欲しい → PCクラスタを自作する**

ワークステーション

自作PCクラスター

ここに挑戦

PCクラスター

大型計算機

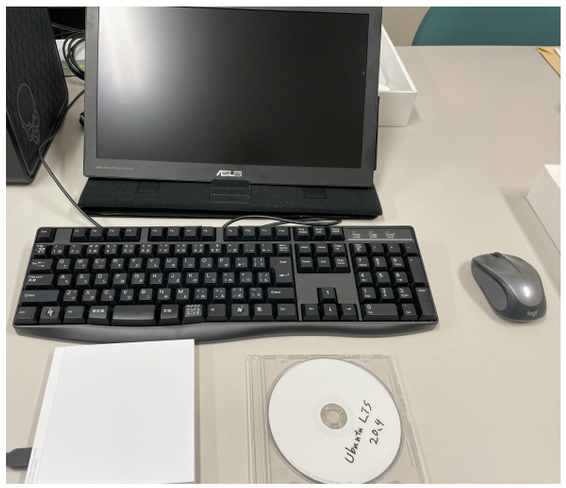
写真：RIKEN

PCクラスター構築ログ

準備した物

- **参考書**
  - 前園 涼 (2017)「自作PCクラスタ超入門」森北出版
- **マシン**
  - XPG GAIA mini PC (GAIAI7G9H630L9-BKCJP) × 3台
    - i7-9750H (6コア)、32GBメモリ、2TB M.2 SSD 搭載
    - 1Gb Ethernet × 2口
    - Thunderbolt 3 × 2口
    - 税込み 12万円/台

- **周辺機器**
  - スイッチングハブ (1 Gbps)
  - LANケーブル
  - 初期設定用キーボード、マウス、ディスプレイ
  - 外付けDVDドライブ
  - OSインストールディスク（Ubuntu sever LTS 20.4）

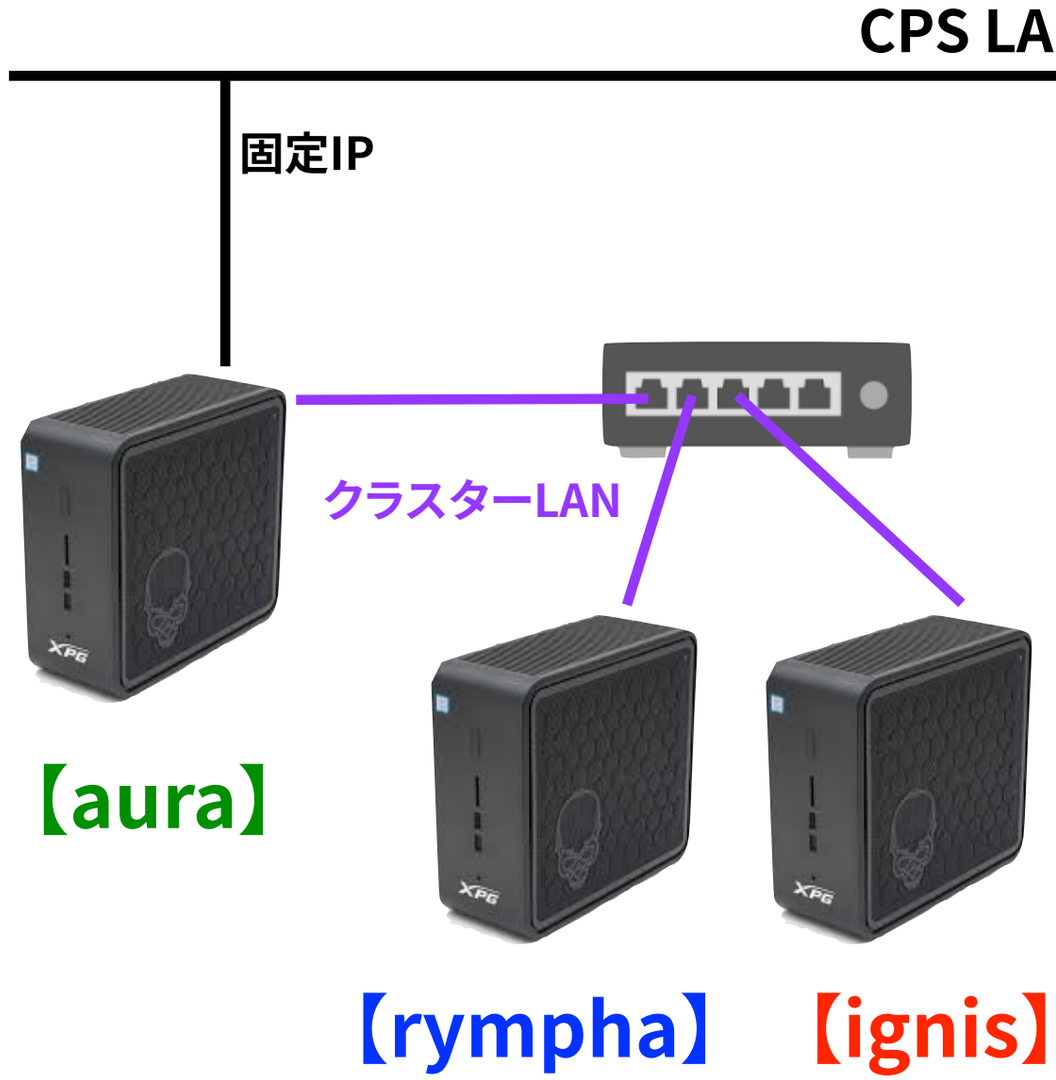


背面

完成予想図

- 1台【aura】(← ホストネーム)
  - 管理ノード 兼
  - 計算ノード 兼
  - ルーター 兼
  - ストレージサーバ

- 残り2台【rympha】【ignis】
  - 計算ノード



手順

1. OSインストール

2. 必要なパッケージの導入

3. ネットワーク設定

4. ノード内並列計算試験

5. ノード間並列計算試験

6. データ領域の作成・共有

7. ジョブ管理システムの導入

✳並列計算試験には、参考書に付属していた、第1原理量子拡散モンテカルロ法 電子状態計算プログラム「CASINO」を用いた。

✳本手順は、おおよそ参考書に従っているが、OSのバージョン違いによる細かな違いがある。

✳作業ログベースなため、無駄な/重複した手順も含まれている（と思う。ちゃんと検証してない）

1. OSインストール【aura, rympha, ignis】

- **OSインストールディスクから起動し、OSをインストール**
  - 最新版のOSをネットワークインストールした
  - sshを導入にチェック
  - hostname はそれぞれ **aura**, **rympha**, **ignis** とした
  - IPアドレスは(とりあえず)DHCPから取得
- **パッケージ更新**
  - sudo apt update
  - sudo apt upgrade
  - sudo reboot
- fish, lv, glances, x11-apps の導入 (必須ではない)
  - sudo apt install fish
  - sudo apt install lv
  - sudo apt install glances
  - sudo apt install x11-apps

【メモ】
なにも考えずにserver版ubuntuをインストールしたので、LVM（Logical Volume Maneger）が有効になった。

2. 必要なパッケージの導入【aura, rympha, ignis】

- **コンパイラ gcc, g++, cpp (ver. 9.4), make**
  - sudo apt install gcc g++ cpp make
- **サーバ環境 ssh, nfs**
  - sudo apt install openssh-server nfs-common
- **並列処理環境 openMP, MPI**
  - sudo apt install libgomp1 openmpi-bin libopenmpi-dev
- **数値計算ライブラリ BLAS, LAPACK**
  - sudo apt install libopenblas-dev libblas-dev liblapacke-dev liblapack-dev
- **描画用 gnuplot** # gtk環境も一緒に導入される
  - sudo apt install gnuplot
- **休止状態に入らないようにする**
  - sudo nano /usr/share/gdm/dconf/90-debian-settings
  - 下記のように編集

```
[org / gnome / settings-daemon / plugins / power]
sleep-inactive-ac-timeout = 0
sleep-inactive-battery-timeout = 0
```

- **再起動**
  - sudo reboot

3. ネットワーク設定【aura, rympha, ignis】

- **ツール導入**
  - sudo apt install net-tools
- **インターフェース確認**
  - sudo ifconfig
- **設定 yamlファイルの設置***
  - cd /etc/netplan/
  - sudo mv 00-installer-config.yaml 00-installer-config.yaml.disabled
  - sudo nano 99-netcfg.yaml

【auraの場合】

```
network:
  version: 2
  ethernets:
    eno1:  # プライベートネットワーク側
      addresses: [192.168.0.1/24]
      nameservers:
        addresses: [XXX.XX.XX.X, XXX.XX.XX.X]
        search: []
      optional: true
    enp111s0: # CPSネットワーク側
      addresses: [XX.XX.X.XXX/24]
      gateway4: XX.XX.X.XXX
      nameservers:
        addresses: [XXX.XX.XX.X, XXX.XX.XX.X]
        search: []
      optional: true
```

【rymphaの場合】

```
network:
  version: 2
  ethernets:
    eno1:  # プライベートネットワーク側
      addresses: [192.168.0.2/24]
      nameservers:
        gateway4: 192.168.0.1
        addresses: [XXX.XX.XX.X, XXX.XX.XX.X]
        search: []
      optional: true
    enp111s0: # CPSネットワーク側
      dhcp4: true
```

【ignisの場合】

```
network:
  version: 2
  ethernets:
    eno1:  # プライベートネットワーク側
      addresses: [192.168.0.3/24]
      nameservers:
        gateway4: 192.168.0.1
        addresses: [XXX.XX.XX.X, XXX.XX.XX.X]
        search: []
      optional: true
    enp111s0: # CPSネットワーク側
      dhcp4: true
```

- **設定反映**
  - sudo netplan apply

*参考URL｜https://linuxfan.info/ubuntu-2004-server-static-ip-address

3. ネットワーク設定 (2) 【aura】

- **auraのルータ化 【auraのみ】**
  - **ルーティングの有効化**
    - sudo nano /etc/sysctl.conf
    - 以下の行を有効化
      `net.ipv4.ip_forward=1`
  - **NAT設定**
    - sudo apt install iptables-persistent
    - sudo su
    - iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o ens160 -j MASQUERADE
    - iptables-save > /etc/iptables/rules.v4

3. ネットワーク設定 (3) 【aura】

- **auraのルータ化 【auraのみ】**
  - **DHCPサーバの設定（多分必須ではない）**
    - パッケージ導入
      sudo apt install isc-dhcp-server
    - 以下をコメントアウト

```
# option domain-name "example.org";
# option domain-name-servers ns1.example.org, ns2.example.org;
```

    - 以下を追記

```
default-lease-time 600;
max-lease-time 7200;
# Added for private network
option routers 192.168.0.1;
option domain-name-servers 192.168.0.1;
option domain-name "lagrange";
subnet 192.168.0.0 netmask 255.255.255.0 {
range 192.168.0.101 192.168.0.199;
}
```

    - 有効化
      sudo systemctl enable isc-dhcp-server.service
      sudo systemctl start isc-dhcp-server.service
  - **DNSサーバの設定** ← 必要ないのでやめた

3. ネットワーク設定 (4) 【aura, rympha, ignis】

- **hostsファイルの編集** # ホスト名でログインできるようにする
  - sudo nano /etc/hosts
  - 以下を追記

```
192.168.0.1 aura
192.168.0.2 rympha
192.168.0.3 ignis
```

  - 既存の自ホスト名はコメントアウトする

4. ノード内並列計算試験【aura】

- **コンパイル**
  - cd setupMaezono/casino
  - setenv CASINO_ARCH linuxpc-gcc-parallel
  - make
- **逐次計算**
  - cd ../ioCasino
  - ../casino/bin_qmc/linuxpc-gcc-parallel/opt/casino
  - out, vmc.hist, config.out が出力される* → **実行時間 : 42.4 sec**
- **並列計算**
  - mpirun -n **2** ../casino/bin_qmc/linuxpc-gcc-parallel/opt/casino --parallel
    - **実行時間 : 21.8 sec**
  - mpirun -n **4** ../casino/bin_qmc/linuxpc-gcc-parallel/opt/casino --parallel
    - **実行時間 : 11.6 sec**
  - mpirun -n **6** ../casino/bin_qmc/linuxpc-gcc-parallel/opt/casino --parallel
    - **実行時間 : 8.2 sec**
  - mpirun -n **8** --oversubscribe ../casino/bin_qmc/linuxpc-gcc-parallel/opt/casino --parallel
    - **実行時間 : 11.3 sec** **・・・ 1ノードのコア数より多いので速くならない**

*これらの出力ファイルは、都度手動で消す。残っていると、次の計算が出来ない。

5. ノード間並列計算試験 (1) 【aura】

- **ファイルのコピー** 【aura】
  - rsync -avz setupMaezono rympha:~/.
  - rsync -avz setupMaezono ignis:~/.　　**全ノード同じディレクトリにコピー**
- **マシンファイルの作成**
  - nano ~/machinefile
  - 以下を記述（ipアドレスとコア数）

```
192.168.0.1 cpu=6
192.168.0.2 cpu=6
192.168.0.3 cpu=6
```

- **実行**
  - cd setupMaezono/ioCasino
  - mpirun -machinefile ~/machinefile -np ../casino/bin_qmc/linuxpc-gcc-parallel/opt/casino --parallel
  - 次のメッセージが出て、計算が終わらない

5. ノード間並列計算試験 (2) 【aura】

```
Open MPI detected an inbound MPI TCP connection request from a peer
that appears to be part of this MPI job (i.e., it identified itself as
part of this Open MPI job), but it is from an IP address that is
unexpected.  This is highly unusual.

The inbound connection has been dropped, and the peer should simply
try again with a different IP interface (i.e., the job should
hopefully be able to continue).

  Local host:           rympha
  Local PID:            3917
  Peer hostname:        aura ([[65043,1],2])
  Source IP of socket: XX.XX.X.XXX
  Known IPs of peer:
       192.168.0.1
```

aura は2つipアドレスを持っているため

- **openMPIの設定ファイルを編集【aura】**
  - sudo nano /etc/openmpi/openmpi-mca-params.conf
  - 末尾に **btl_tcp_if_include=eno1** を追加。
  - mpirun -machinefile ~/machinefile -n **8** ../casino/bin_qmc/linuxpc-gcc-parallel/opt/casino --parallel
    - **実行時間 : 6.3 sec**
  - mpirun -machinefile ~/machinefile -n **16** ../casino/bin_qmc/linuxpc-gcc-parallel/opt/casino --parallel
    - **実行時間 : 3.4 sec**
  - なお、上記ファイルを変更したくない場合は、実行時のオプションで --mca btl_tcp_if_include eno1 をつける。

★ **ここまでの設定で、ノード間並列計算が可能!**

6. データ領域の作成・共有

毎度、計算ノードにプログラムや設定ファイルをコピーするのは面倒

aura にデータ領域を作成し

そのデータ領域を rympha・ignis がネットワークマウントする

6. データ領域の作成・共有 (1) 【aura】

初期状態は、2TBのSSDが
- VG Name: ubuntu-vg # 論理ボリュームグループ
- LV Name: ubuntu-lg # 論理ボリューム（ / としてマウント）

- **論理ボリュームグループから、新しい論理ボリュームを作成**
  - sudo lvcreate --name data --size 1.5TB ubuntu-vg
    論理ボリューム名：data　大きさ: 1.5TB
- **作成した論理ボリュームに ext4 ファイルシステムを作成し、マウントする**
  - sudo mkfs.ext4 /dev/ubuntu-vg/data
  - sudo mkdir /mnt/data-aura
  - sudo mount /dev/ubuntu-vg/data /mnt/data-aura
  - df -h
- **自動マウントするように fstab に登録**
  - sudo blkid | grep /dev/mapper
    UUIDをコピーする
  - sudo nano /etc/fstab
  - 以下を追記
    UUID="XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" /mnt/data-aura   ext4 defaults 0 0
  - sudo reboot
- **エイリアス作成**
  - ln -s /mnt/data-aura ~/.
- **オーナー・グループ変更**
  - sudo chown hiroki /mnt/data-aura
  - sudo chgrp hiroki /mnt/data-aura

参考：https://qiita.com/TsutomuNakamura/items/93c6333c8dd32aeb197a

6. データ領域の作成・共有 (2)

NFSによるdata領域の共有

- **NFSサービスの設定【aura】**
  - **必要なパッケージの導入**
    - sudo apt install nfs-kernel-server
  - **exportsファイルの編集**
    - sudo nano /etc/exports
    - 以下を追記
      ```
      /mnt/data-aura 192.168.0.0/255.255.255.0(rw)
      ```
  - **NFSの再起動**
    - sudo /etc/init.d/nfs-kernel-server restart
- **NFSマウントの設定【rympha, ignis】**
  - sudo mkdir /mnt/data-aura
  - sudo nano /etc/fstab
  - 以下を追記
    ```
    # NFS mount data-aura
    192.168.0.1:/mnt/data-aura /mnt/data-aura nfs defaults  0     0
    ```
  - sudo mount -a
  - ln -s /mnt/data-aura ~/.

参考: https://qiita.com/TsutomuNakamura/items/93c6333c8dd32aeb197a

7. ジョブ管理システムの導入

ノード数・コア数が多くなると、計算資源の管理
（どのプロセスがどのノードの何コアをどれくらいの時間使うか）が大変

これらを一元管理するツールとして、
ジョブ管理システム*を導入する

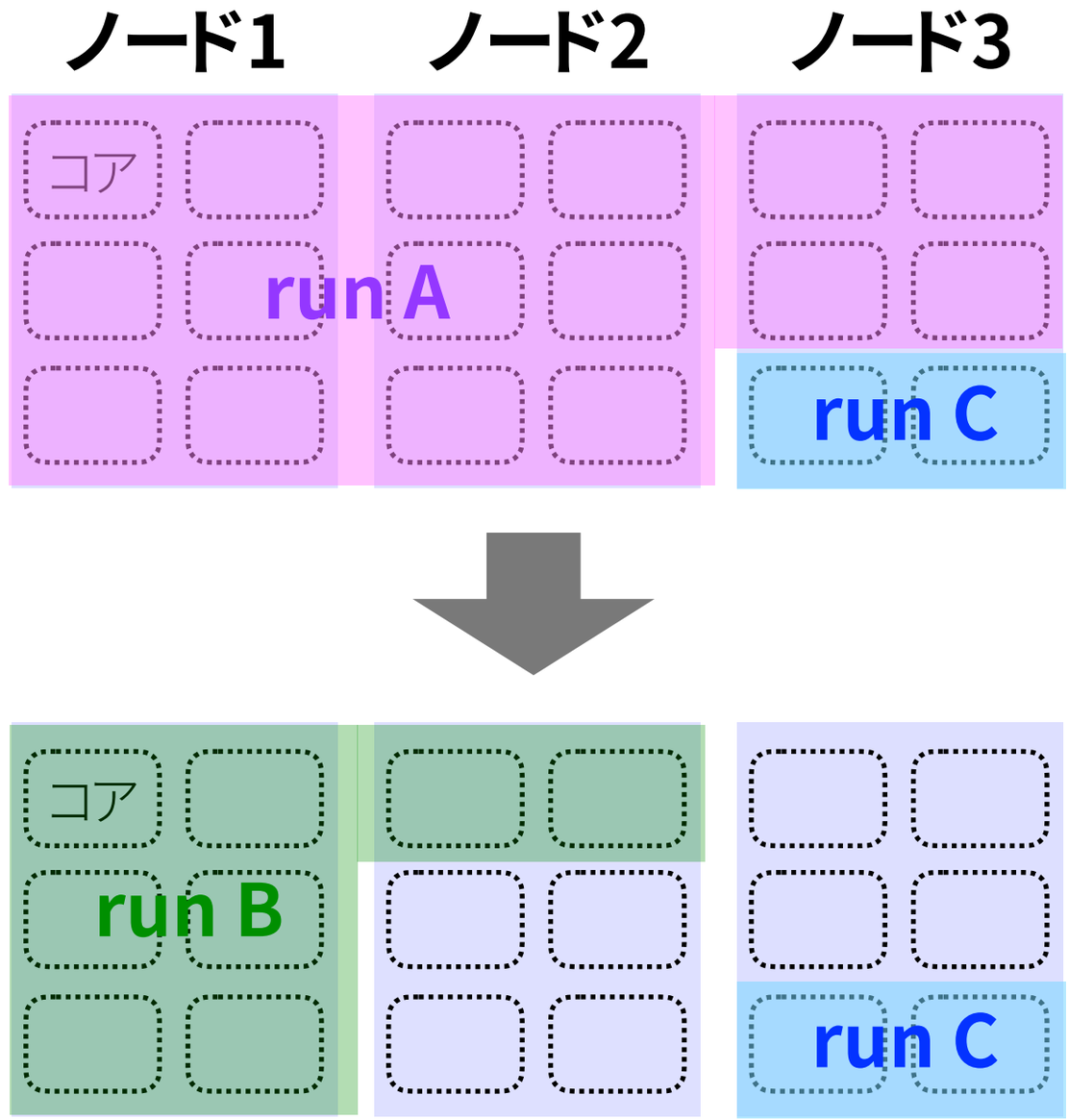
*広義にはcronなどもジョブ管理システムに含まれるらしいが、ここでは多ノード並列計算機のプロセス管理ツールのこと。
ジョブスケジューラやバッチジョブシステムとも呼ばれる

ジョブ管理システムの概要

- 計算に必要な並列数・実行時間は様々
- 例えば、全18コアのシステムで
  以下の3つの計算を実行したいとき
  - **run A** : 16並列・2時間
  - **run B** : 8並列・6時間
  - **run C** : 2並列・12時間
- まず **run A** と **run C** で計算を始めて、
  run A が終わったら、**run B** を開始すると
  計算資源が効率的に使える

➡ 複数プロセスをシステム内に自動で配置して
計算資源の空転をなるべく減らす

- 複数人で使用する際、最長実行時間に制限
  をかけることで、1人で長時間占有することを
  防げる



ジョブ管理システムの例

- **PJM**
  - 「富岳」、九大ITOなど、研究機関・大学の大型計算機で採用
  - 富士通製？

- **TORQUE**
  - オープンソースのジョブ管理システム
  - セット売りのPCクラスターで導入されていた
  - 昔は apt でインストールできたが、今はできない

- **Slurm**
  - オープンソースのジョブ管理システム
  - 多くの大型計算機で採用されている
  - apt でインストール可能 ・・・ 今回はこれを採用

7. ジョブ管理システムの導入 (1)

- **認証ソフト munge の導入【aura, rympha, ignis】・・・**ノード間のアクセスを可能にする
  - sudo apt install munge
  - sudo systemctl start munge
  - systemctl status munge.service
  - id munge
    →ユーザ munge が作成されている
- **aura の認証キーを rympha, ignis にコピーする**
  - 【aura】sudo rsync -a /etc/munge/munge.key hiroki@rympha:.
  - 【aura】sudo rsync -a /etc/munge/munge.key hiroki@ignis:.
  - 【rympha, ignis】sudo mv munge.key /etc/munge/.
  - 【rympha, ignis】sudo chown munge /etc/munge/munge.key
  - 【rympha, ignis】sudo chgrp munge /etc/munge/munge.key
  - 【rympha, ignis】sudo systemctl restart munge
- **認証の確認 【aura】**
  - munge -n | ssh rympha unmunge
  - munge -n | ssh ignis unmunge
    →STATUS: Success (0) が出たら成功

参考：http://www-fps.nifs.ac.jp/ito/memo/slurm01.html
https://qiita.com/JeJeNeNo/items/c545e72373d4d1deb36a
https://mumeiyamibito.0am.jp/分子シミュレーション関連/環境構築/slurm

7. ジョブ管理システムの導入 (2)

- **Slurm の導入・設定**
  - 【aura】 sudo apt install slurm-wlm
  - 【rympha, ignis】 sudo apt install slurm-client
- **設定ファイル slurm.conf の作成【aura】**
  - sudo nano /etc/slurm-llnl/slurm.conf
    http://www-fps.nifs.ac.jp/ito/memo/slurm01.html を参考に作成

/etc/slurm-llnl/slurm.conf

```
====================
# slurm.conf file generated by configurator easy.html.
# Put this file on all nodes of your cluster.
# See the slurm.conf man page for more information.
#
ClusterName=vox-machines ##ドメインとは独立に自由な名前で良いらしい##
ControlMachine=aura       ##管理サーバーのホスト名##
#ControlAddr=
#
SlurmUser=slurm
SlurmctldPort=6817
SlurmdPort=6818
AuthType=auth/munge
#StateSaveLocation=/var/spool
StateSaveLocation=/tmp
#SlurmdSpoolDir=/var/spool/slurmd
SlurmdSpoolDir=/tmp/slurmd


#MailProg=/bin/mail
MpiDefault=none
#MpiParams=ports=#-#
ProctrackType=proctrack/pgid
ReturnToService=1   ##計算ノードの状態更新の為に1が良いらしい##
SlurmctldPidFile=/var/run/slurmctld.pid
SlurmdPidFile=/var/run/slurmd.pid
#SlurmdUser=root
SwitchType=switch/none
TaskPlugin=task/none
#
#
# TIMERS
#KillWait=30
#MinJobAge=300
#SlurmctldTimeout=120
#SlurmdTimeout=300
#
#
# SCHEDULING
FastSchedule=1
SchedulerType=sched/backfill
#SchedulerPort=7321
#SelectType=select/linear ##一つの計算ノードでは1つのジョブしか実行させたたくない場合(ノードを跨いだ並列計算は可能)##
SelectType=select/cons_res ##一つの計算ノードにて複数のジョブを実行させたい場合(ノードを跨いだ並列計算も可能)##
SelectTypeParameters=CR_CPU ##後者の場合のjobの割り振り方、CPU単位か、socket単位かなど##
#
#
# LOGGING AND ACCOUNTING
#AccountingStorageType=accounting_storage/none
#JobAcctGatherFrequency=30
#JobAcctGatherType=jobacct_gather/none
#SlurmctldDebug=3
#SlurmctldLogFile=
#SlurmdDebug=3
#SlurmdLogFile=
#
#
# COMPUTE NODES
##計算ノードの登録##
NodeName=aura Sockets=1 CoresPerSocket=6 \
 ThreadsPerCore=2 RealMemory=31818 State=UNKNOWN \

NodeName=rympha Sockets=1 CoresPerSocket=6 \
 ThreadsPerCore=2 RealMemory=31818 State=UNKNOWN \

NodeName=ignis Sockets=1 CoresPerSocket=6 \
 ThreadsPerCore=2 RealMemory=31818 State=UNKNOWN

##ジョブパーティションの登録
PartitionName=vox Nodes=aura,rympha,ignis Default=YES \
 MaxTime=INFINITE State=UP OverSubscribe=NO \
 MaxCPUsPerNode=6
# MaxCPUsPerNodeを設定しないと、Hyperthreading込みでジョブを実行する。
====================
```

詳細分かってないけど

7. ジョブ管理システムの導入 (3)

- **起動【aura】**
  - sudo systemctl start slurmctld.service
  - sudo systemctl restart slurmctld.service
  - sudo systemctl start slurmd.service
  - sudo systemctl restart slurmd.service
- **確認【aura】**
  - systemctl status slurmctld.service
  - systemctl status slurmd.service
- **設定ファイルの配布【aura】**
  - scp /etc/slurm-llnl/slurm.conf rympha:.
  - scp /etc/slurm-llnl/slurm.conf ignis:.
- **設定ファイルのコピー【rympha, ignis】**
  - sudo cp slurm.conf /etc/slurm-llnl/.
- **起動【rympha, ignis】**
  - sudo systemctl start slurmd.service
  - sudo systemctl restart slurmd.service
- **確認【rympha, ignis】**
  - systemctl status slurmd.service

ジョブ投入テスト【aura】

- **ジョブスクリプトの作成**
  - cd ~/data-aura/setupMaezono/ioCasino
  - nano job.sh
    下記を作成

```
==================
#!/bin/bash
#SBATCH --get-user-env
#SBATCH -A hiroki
#SBATCH -p vox
#SBATCH -J CASINO
#SBATCH -N 3           #num of node
#SBATCH -n 16          #num of total mpi processes
#SBATCH -c 1           #num of threads per mpi processes
#SBATCH --ntasks-per-node=6  #num of processes per node
#SBATCH -o log
#SBATCH -e err
##SBATCH -t 01:10:00

mpirun -np 16 ../casino/bin_qmc/linuxpc-gcc-parallel/opt/casino --parallel
==================
```

- **ジョブ投入**
  - sbatch job.sh
    → Submitted batch job 数字　が表示される

参考：https://www.j-focus.jp/user_guide/ug0004020000/

Slurmコマンド

- **ジョブ投入**
  - sbatch job.sh

- **実行確認**
  - squeue

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
    2       vox   CASINO   hiroki  R       0:08      3 aura,ignis,rympha
```

- **ノード状況確認**
  - sinfo

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
vox*         up   infinite      3   idle aura,ignis,rympha
```

- **ジョブキャンセル**
  - scancel ジョブID

まとめ

- 数値計算・シミュレーションは、さまざまな学問分野で利用されている
- 計算機の能力が高いほど **より速く、より多くの、より大規模な** 計算ができる

- 今、CPUは多コアの時代。クロック数は伸びない
  - 1CPU当たりの最大 64 コア（今のところ）
  - 多コアのCPUは高価

➡ PCクラスターを構築してみた

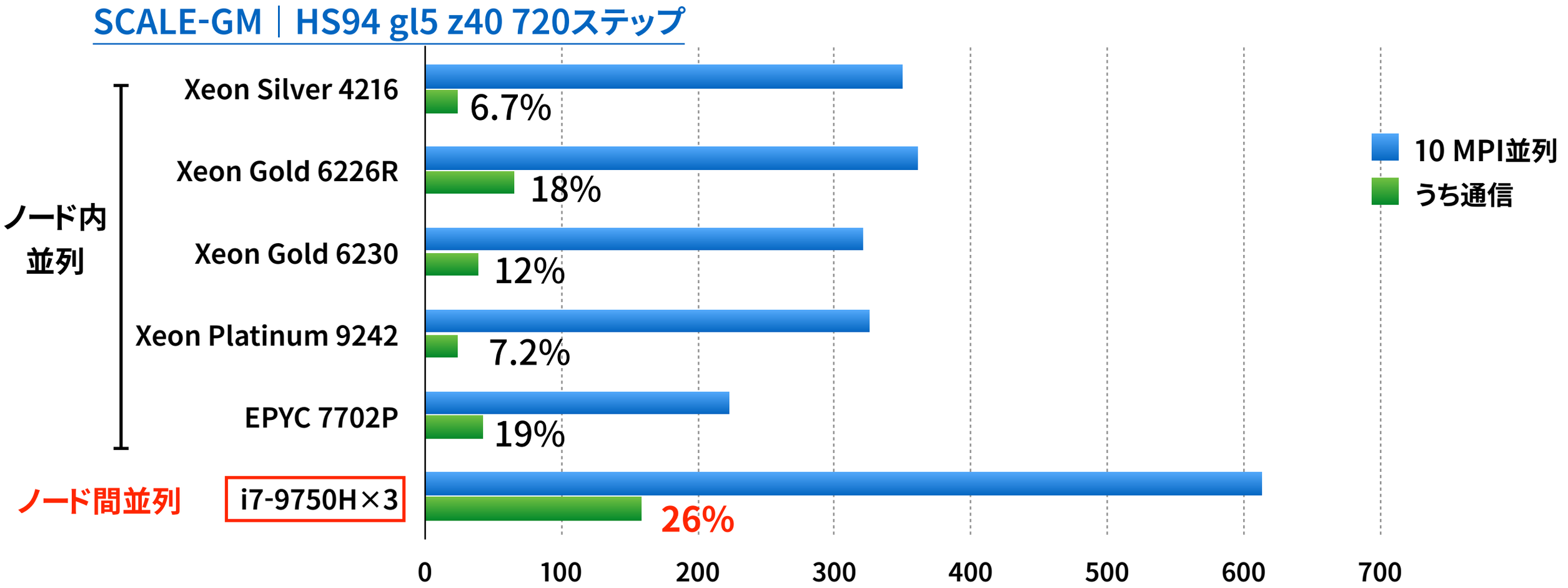
1. OS インストール
2. 必要なパッケージの導入
3. ネットワーク設定
4. ノード内並列計算試験
5. ノード間並列計算試験
6. データ領域の作成・共有
7. ジョブ管理システムの導入

購入費用 : 約 36 万円

同コア数・同メモリ量のWS*
参考価格 : 約 68 万円

*システムワークスPOWER MASTER Vision A9467/Xeon W-2295/16GB×6/2TBSSD

おまけ｜全球大気力学モデルでの性能は？



- 今回の自作PCクラスター遅い原因
  - ノード間の通信が 1 Gbps なので、通信に時間がかかっている
  - メモリ帯域幅が狭い（2 チャネル）
  - キャッシュが少ない（12MB）

おまけ｜Thunderbolt 3 で通信?

- **IP over thunderbolt という仕組みがある**
  - Mac同士の高速通信 (40 Gbps)が可能

- **パッケージの導入**
  - sudo apt install thunderbolt-tools

- netplan に thunderbolt0 の設定を追加すると、IPが与えられ、開通する
- ただし、
  - iperf3 で計測する速度が、
    - ignis → rympha は速い（~12 Gbps）が
    - rympha → ignis はとても遅い（~10 Mbps）

  - ファイルの読み書きを伴う転送を行っていると、マシンがフリーズする。
    - mpi並列しようとしてもフリーズする **→ 作戦失敗**

参考：https://applech2.com/archives/40469310.html