Parallel jobs and benchmarking

Jerez Te
NIH HPC
January 16, 2018

Outline

Parallel Computing

- Slurm and Parallel Computing

- NIH HPC Policies and Tips

Benchmarking

- Molecular dynamics jobs

- Genomics jobs

- Spark (distributed) & deep learning jobs

***Disclaimer: focus is on efficiency. Other factors: accuracy, features, compatibility in pipeline

Parallel Computing

Many calculations or execution of processes carried out simultaneously

(-Wikipedia)

Cluster Basics

computer = node = blade

=

cpus
memory
disk
network adapter

cluster of 13 nodes

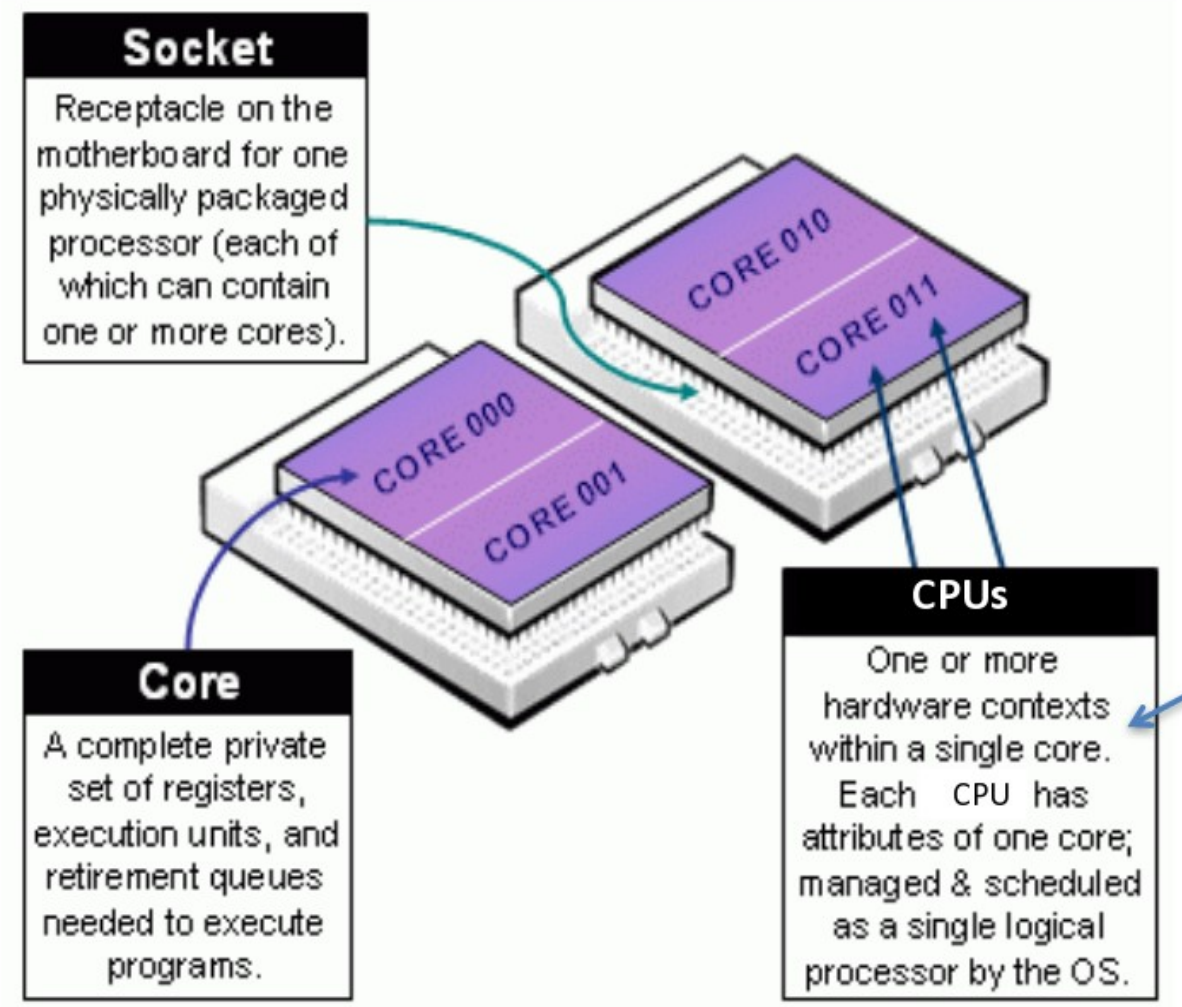
fileserver (disk storage)

network switch

Some apps can use multinodes

- Nodes communicate: Message parsing interface (MPI) allows parallel programming on a variety of computer hardware. Needs a copy of data (not suited for large data).

Multi-threading



Multiple independent threads within the context of a single process. Using multi-core processors with shared memory.

e.g., --cpus-per-task=8; --ntasks=1 --nodes=1 (multiple CPUs, single task)

Example: 8-CPU node

Multi-threading vs. MPI parallism

HPC @ NIH

High-Performance Computing at the NIH

Search:

Status | Applications | Reference Data | Storage | User Guides | User Dashboard | How To | About

bowtie2 on Biowulf2 & Helix

Description

Bowtie2 is a fast, multi-threaded, and memory efficient aligner for short read sequences. It uses an FM index to achieve a moderate memory footprint of 2 - 4 GB, depending on genome size and alignment parameters. Performance scales well with thread count.

Quick Links

- bowtie2 on Helix
- Serial bowtie2 on Biowulf2
- Swarm of bowtie2 jobs
- Interactive bowtie2 on Biowulf2

```
#! /bin/bash
set -o pipefail
set -e

echo "Running on $SLURM_CPUS_PER_TASK CPUs"
threads=$(( SLURM_CPUS_PER_TASK - 2 ))
module load bowtie/2-2.2.5 || exit 1
module load samtools/1.2 || exit 1

cd /data/$USER/test_data
export BOWTIE2_INDEXES=/fdb/igenomes/Mus_musculus/UCSC/mm9/Sequence/Bowtie2Index
bowtie2 --sensitive-local -p ${threads} --no-unal -x genome \
      -U /usr/local/apps/bowtie/TEST_DATA/ENCFF001KPB.fastq.gz \
   | samtools view -q30 -Sb - > ENCFF001KPB.bam
```

```
sbatch --cpus-per-task=16 --mem=5g example_bowtie2_slurm.sh
```

Resource to know what the app is:

https://hpc.nih.gov/apps/

Making efficient use of Biowulf's multinode partition

- Users can use up to 6,272 CPUs at one time or 12,544 CPUs for time limit < 8 hours (qos="turbo"). Command: **batchlim**

- The NIH HPC staff will ask to see proof that jobs requesting more than **512 CPUs** are actually able to take advantage of them.

- Multinode gpus limit = **16 gpus**

  4 k80 nodes

  8 k20x nodes

https://hpc.nih.gov/policies/multinode.html

Tips

- Don't use multinode for multi-threaded jobs

- Use homogeneous resources

sbatch --partition=multinode **--constraint=x2650** --ntasks=64 --ntasks-per-core=1 --time=168:00:00 --exclusive jobscript

- Benchmark your job

- Consider storage and memory requirements

Overallocation of memory or walltime may lead to longer wait time

https://hpc.nih.gov/policies/multinode.html

Efficiency

Efficiency = (work done for N CPUs)/(N * work for 1 CPU)

- For example, if you double the number of CPUs you use, but the runtime of your job is only shortened by 30%, that is not a good use of resources.
- Please note: it is in your own best interest to run with a number of cores that has a parallel efficiency above 0.7, since job priority is based on the amount of past CPU usage over the last few months.

Do's and don't for parallel jobs

Resource	Do	Don't
Number of CPUs	Benchmark your job as far as possible to determine a reasonable number of CPUs to run on for good parallel efficiency.	Request the maximum possible number of CPUs without knowing that your job will scale. The HPC staff will ask submitters of large jobs (more than 512 CPUs) to demonstrate scaling. In addition, larger jobs will usually wait longer in the queue before starting.
Wall times	Break your work down into the smallest reasonable chunk size. Request a wall time based on benchmarking that covers how long you expect the job to run for plus a 15%-25% buffer. If benchmarking cannot give an accurate wall time for a production run, run a single production job and use the amount of time it took plus a buffer as the wall time for similar jobs.	Submit all jobs with a wall time of 10 days. Jobs with longer wall times cannot be scheduled as efficiently, and thus they will wait longer in the queue when the system is busy.
Memory	Use memory utilization from small benchmark jobs to guide resource requests of future jobs. Use jobload and jobhist to monitor utilization.	Just guess at how much memory your job needs. Jobs that exceed their memory limits will be killed by the batch system. Conversly, requesting much more memory than needed will make it harder to schedule your job and may delay its start.
I/O	Start small with jobs that read and write a lot of data and scale them up gradually when you know that they are running well and your disk quota is sufficient to accommodate any new data.	Submit a large parallel job that writes lots of files without testing that the storage system will be able to absorb the I/O.
Other assistance	Ask the HPC staff for assistance if you are not sure what the best strategy for running your workload is, or if you have questions about how to construct a reasonable resource specification.	Plow on ahead without knowing whether what you are doing is making good use of a shared resource.

https://hpc.nih.gov/policies/multinode.html

Benchmarking

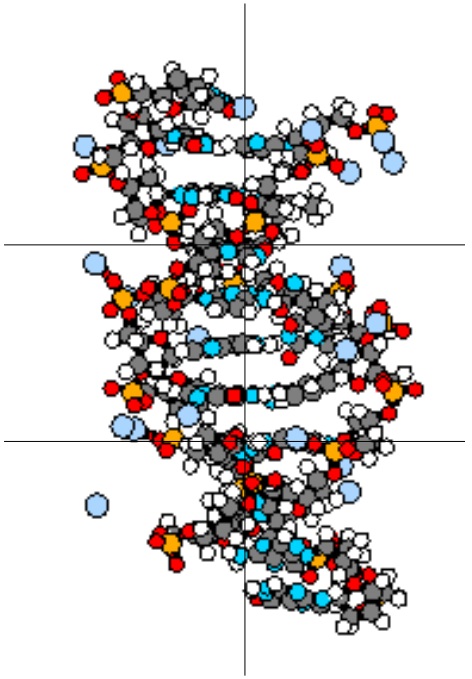
MD simulations

What is MD simulation?

Study of trajectories of atoms for a system of interacting molecules.

Applications:

-study motion of proteins and nucleic acids
-ligand docking (drug design)

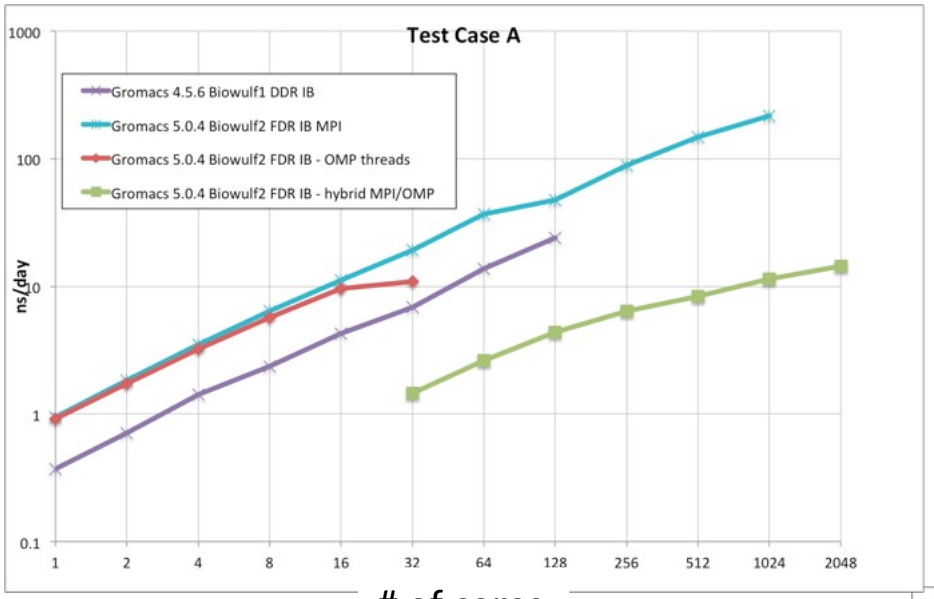


NIH HPC Apps:

- **NAMD**
- **GROMACS**
- **OpenMM**
- CHARMM
- AMBER

MD: Gromacs on CPUs

Better

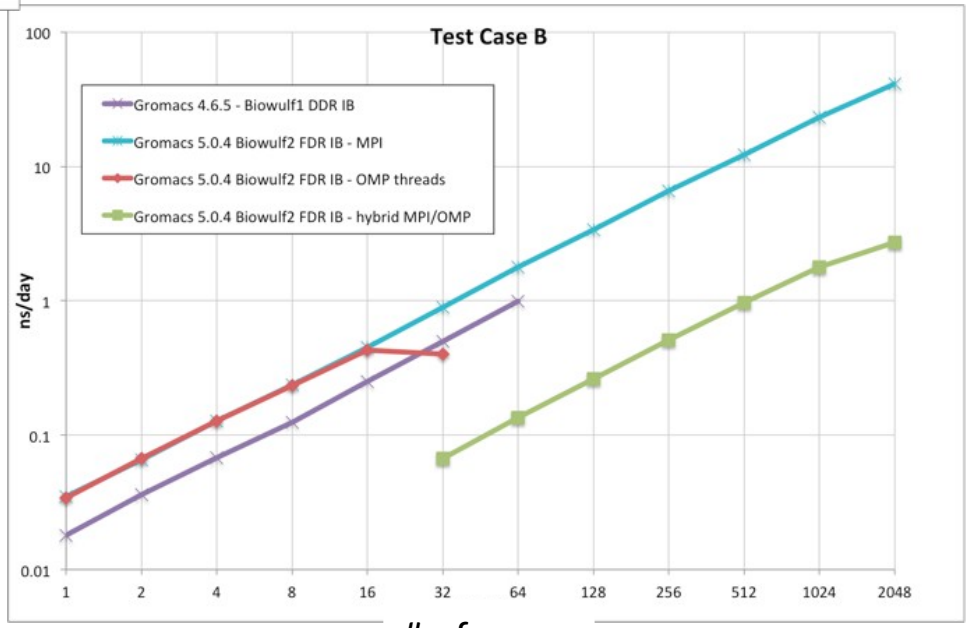


# of cores

Using Gromacs benchmarks

Ion channel system (GluCl protein, 150K atoms)

Cellulose and lignocellulosic biomass (3.3M atoms)



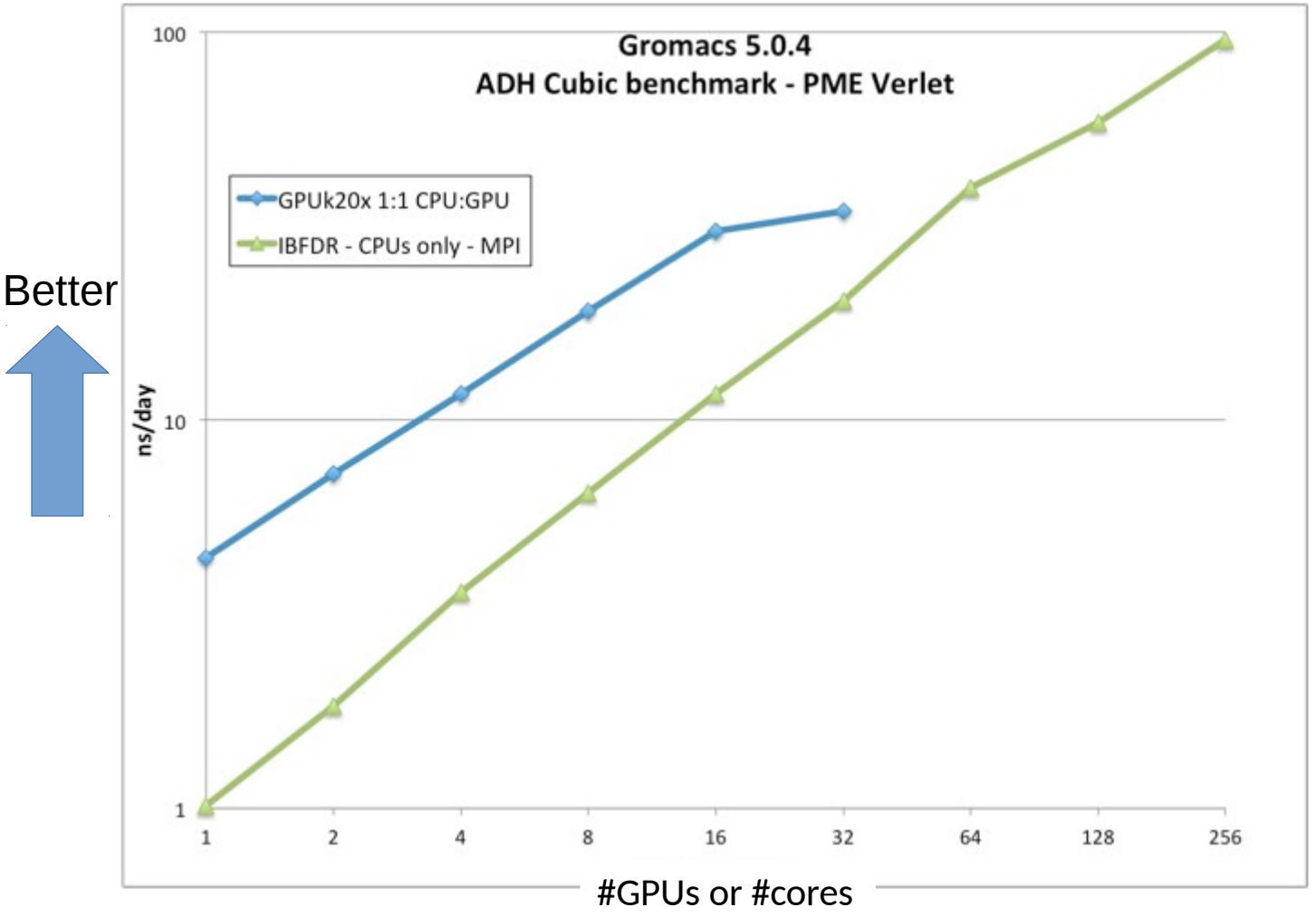
# of cores

sbatch **--partition=multinode** --constraint=x2695 --ntasks=112 --ntasks-per-core=1 --time=24:00:00 –exclusive gromacs_script.sh

*** 4 nodes * 28 cores = 112

https://hpc.nih.gov/apps/gromacs/index.html

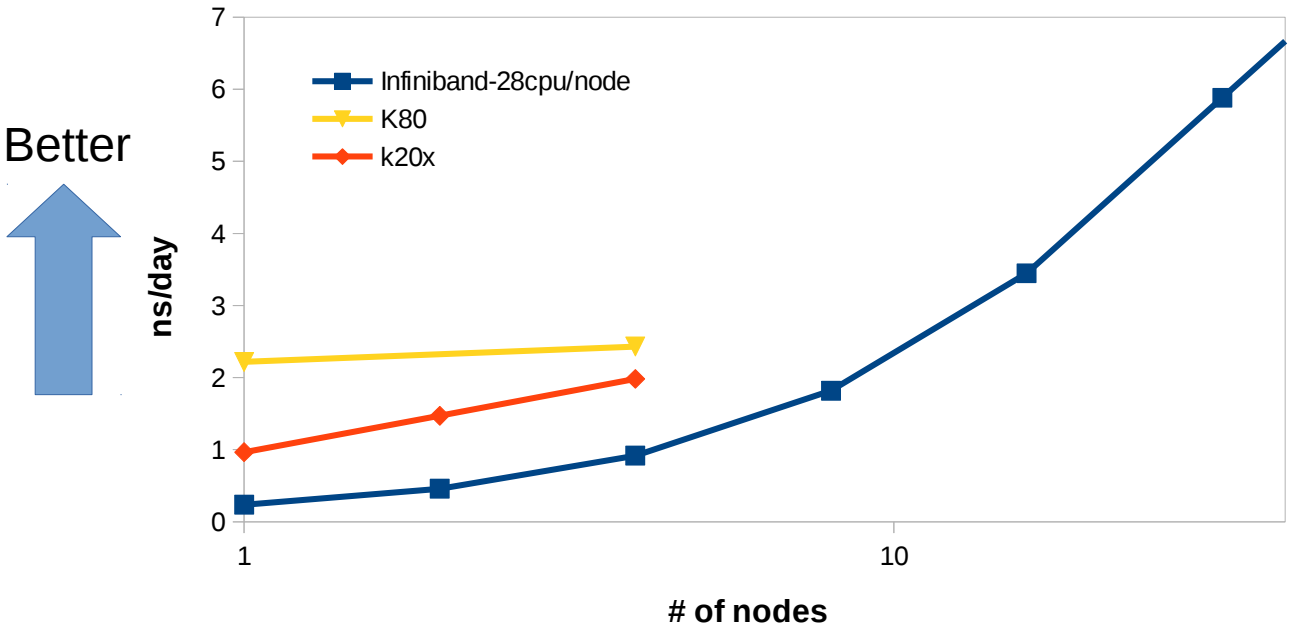
MD: Gromacs on GPUs



Using Gromacs benchmarks

Limit of 16 gpus

MD: NAMD

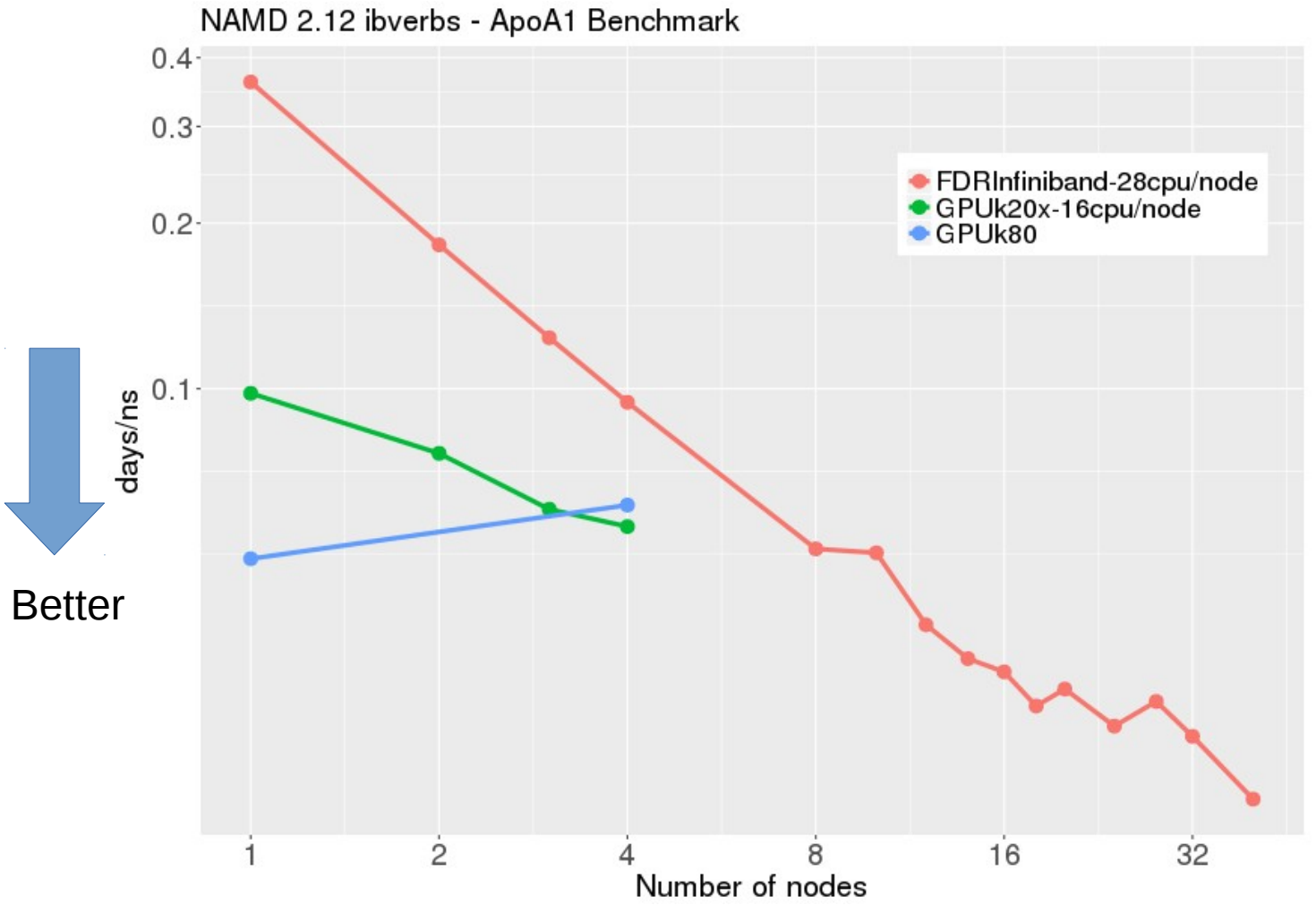


Using NAMD benchmarks

STMV benchmark (1M atoms)

https://hpc.nih.gov/apps/namd/

MD: NAMD



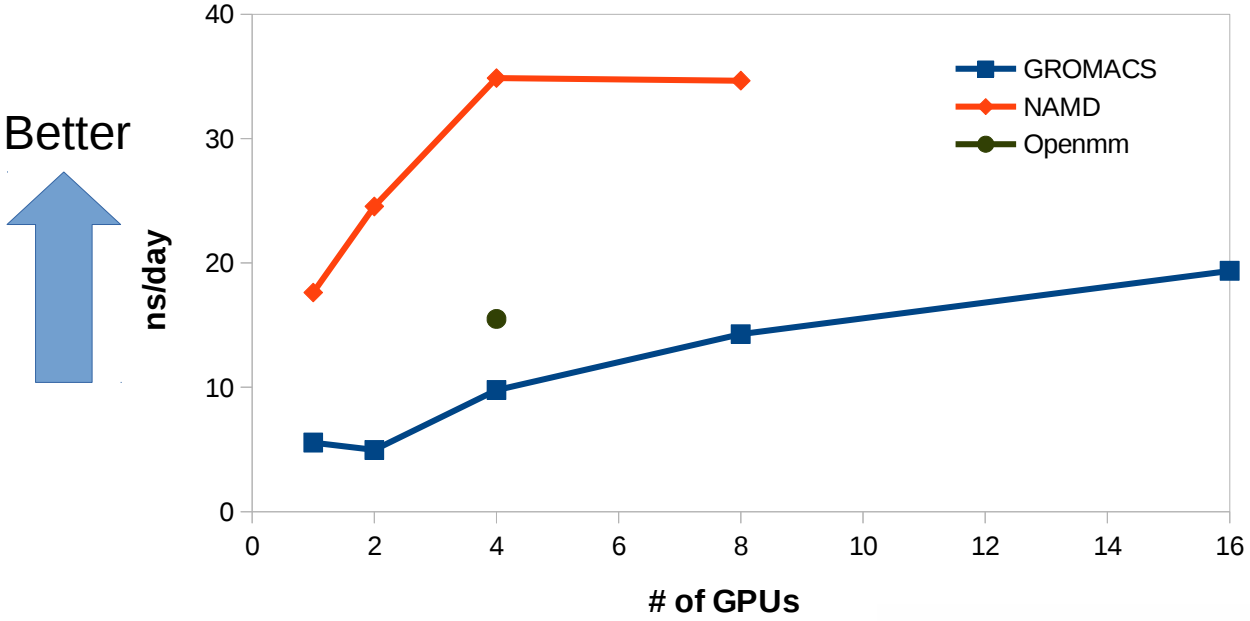
Better

Using NAMD benchmarks

Apoa1 (90K atoms)

Worse performance for multinode gpus in smaller systems

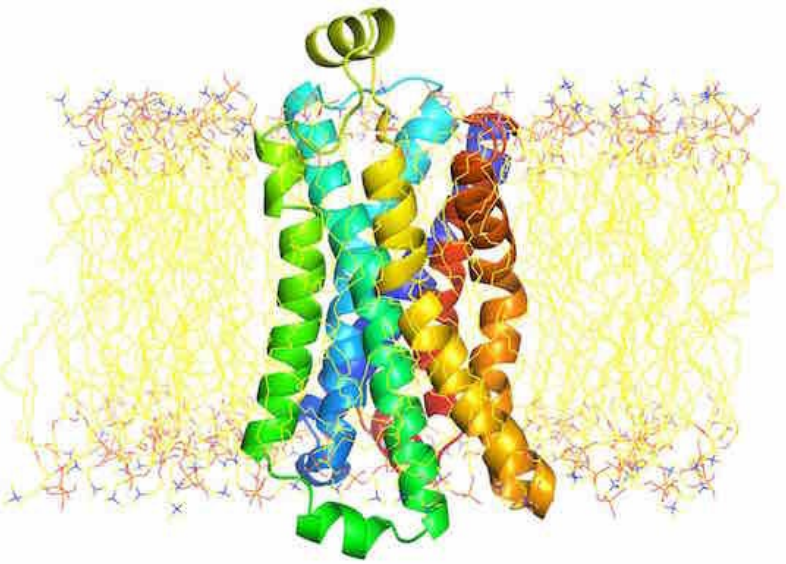
MD simulations: Comparison



Openmm:

- Optimized for GPUs
- GPU-aware
- Recommended 1 gpu

Test case:
- 45,077 atoms
- B2-adrenergic receptor in POPC lipids



http://openmm.org/tutorials/b2ar_membrane/

Monitoring

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.26                 Driver Version: 375.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           On   | 0000:83:00.0     Off |                  Off |
| N/A   73C    P0    71W / 149W |    102MiB / 12205MiB |     33%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K80           On   | 0000:84:00.0     Off |                  Off |
| N/A   46C    P0    86W / 149W |    102MiB / 12205MiB |     36%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K80           On   | 0000:8A:00.0     Off |                  Off |
| N/A   78C    P0    85W / 149W |    120MiB / 12205MiB |     52%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla K80           On   | 0000:8B:00.0     Off |                  Off |
| N/A   61C    P0    91W / 149W |    121MiB / 12205MiB |     50%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0     39676    C   .../_2.12_Linux-x86_64-ibverbs-smp-CUDA/namd2   100MiB |
|    1     39676    C   .../_2.12_Linux-x86_64-ibverbs-smp-CUDA/namd2   100MiB |
|    2     39675    C   .../_2.12_Linux-x86_64-ibverbs-smp-CUDA/namd2   118MiB |
|    3     39675    C   .../_2.12_Linux-x86_64-ibverbs-smp-CUDA/namd2   119MiB |
+-----------------------------------------------------------------------------+
```

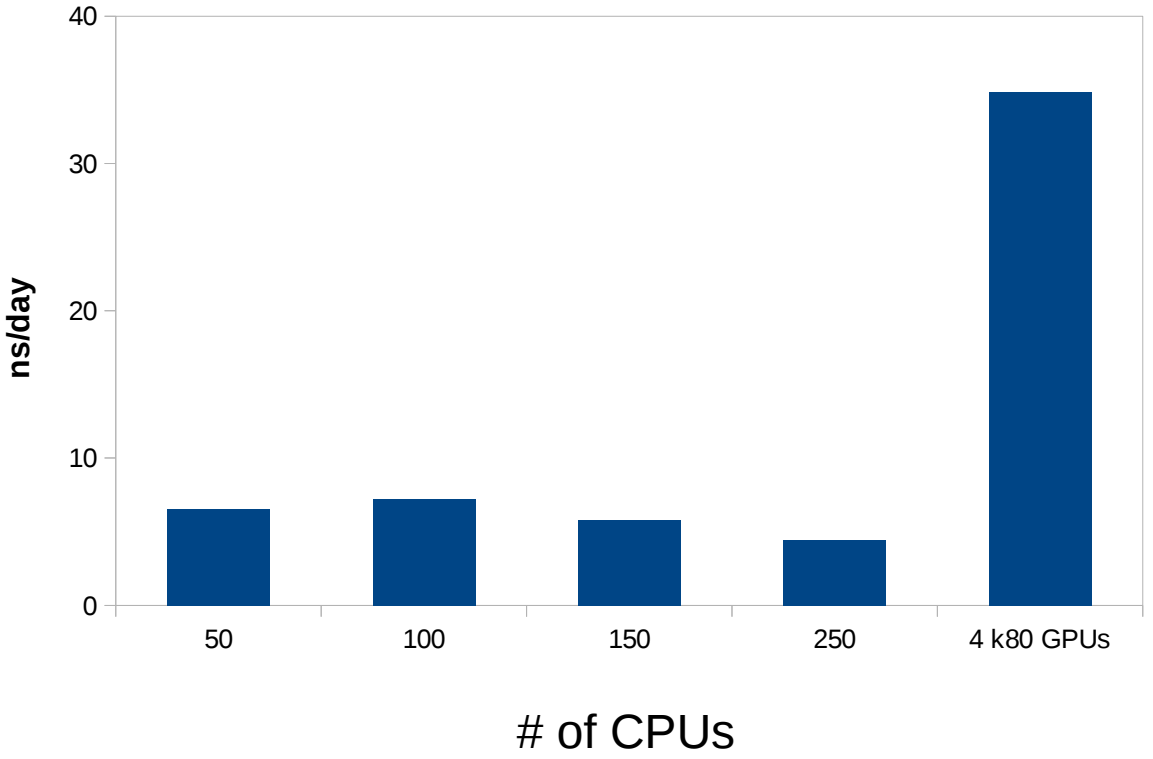
Monitoring is important:
rsh cn4200 nvidia-smi
rsh cn4201 nvidia-smi

MD simulations: NAMD

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.26                 Driver Version: 375.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           On   | 0000:83:00.0     Off |                  Off |
| N/A   32C    P8    26W / 149W |      0MiB / 12205MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K80           On   | 0000:84:00.0     Off |                  Off |
| N/A   26C    P8    32W / 149W |      0MiB / 12205MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K80           On   | 0000:8A:00.0     Off |                  Off |
| N/A   34C    P8    25W / 149W |      0MiB / 12205MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla K80           On   | 0000:8B:00.0     Off |                  Off |
| N/A   31C    P8    32W / 149W |      0MiB / 12205MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Knights Landing: NAMD

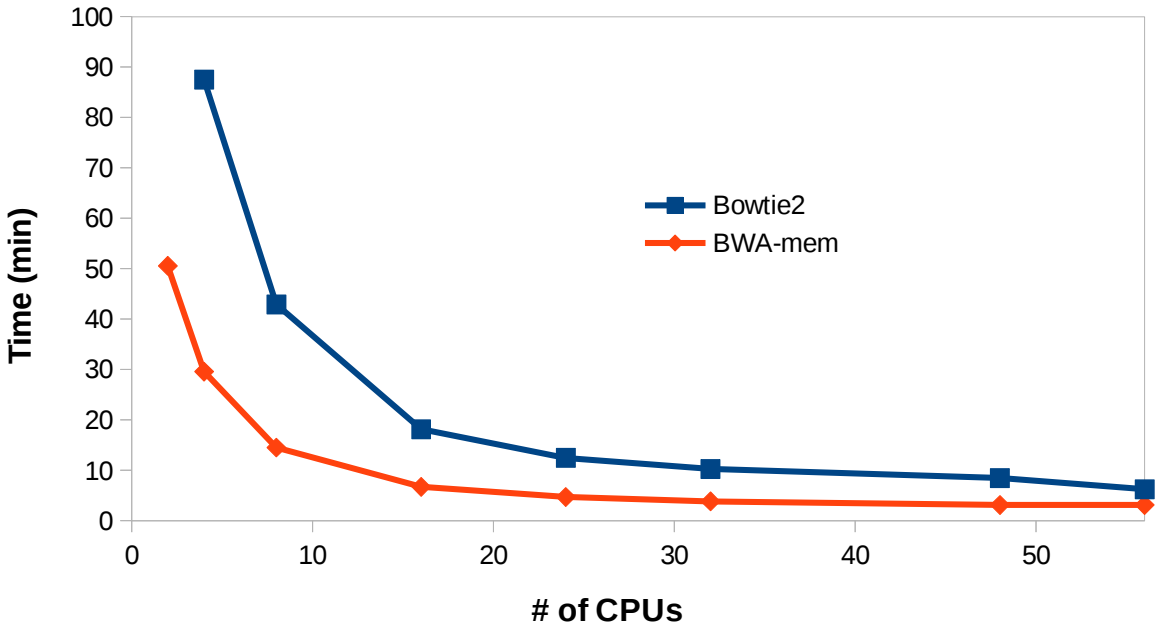


Bioinformatics/Genomics

- Read alignment: Bowtie2 and Bwa
- Spliced read alignment: STAR and Hisat2
- Tools: Samtools and Sambamba
- WGS: Isaac4, Strelka, Canvas

***Disclaimer: focus is on efficiency. Other factors: accuracy, features, compatibility in pipeline

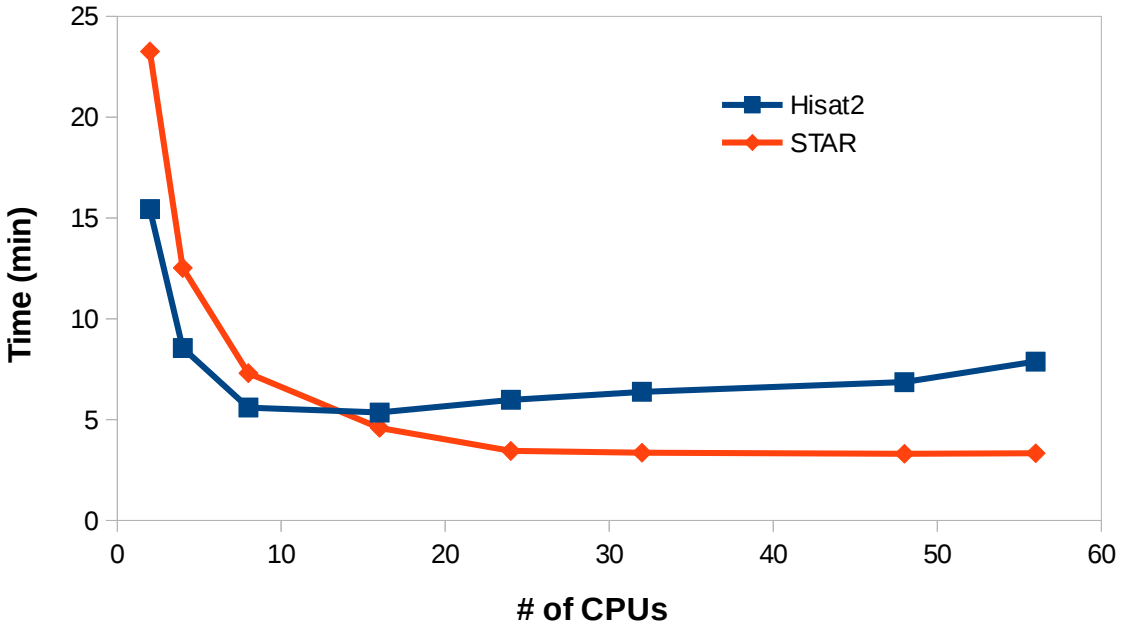
Read alignment: Bowtie2 and Bwa



SRR2556952
Unzipped fastq ~13gB (zipped – 2.5gB)
Output: unsorted sam file

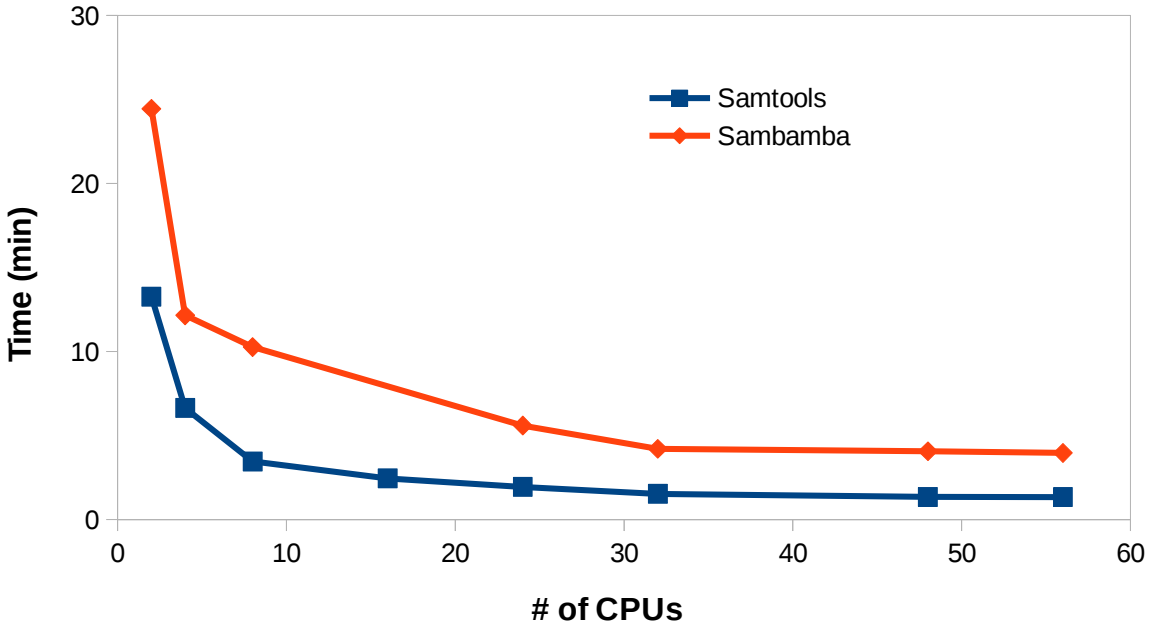
sbatch --cpus-per-task=48 --mem=30g --constraint=x2680 submit.sh

RNA-seq: STAR and Hisat2 (spliced read alignment)



SRR2556952
Unzipped fastq ~13gB (zipped – 2.5gB)
Output: unsorted sam file

Samtools and Sambamba (filtering and sorting)



Using the sam file from Hisat2
- Filter Q-score<30
- Sort
- Bam output

Whole genome sequencing: Isaac4

Workflow:

Fastq files

↓

Isaac4 – Alignment

↓

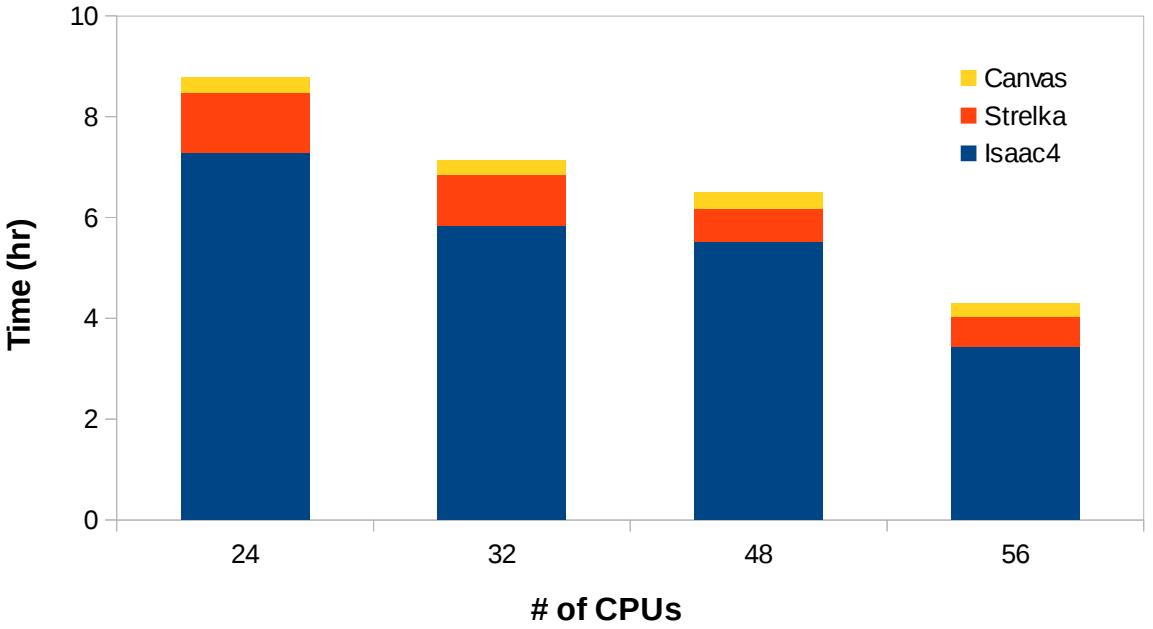
Strelka – Variant calling

↓

Canvas – Copy number variations

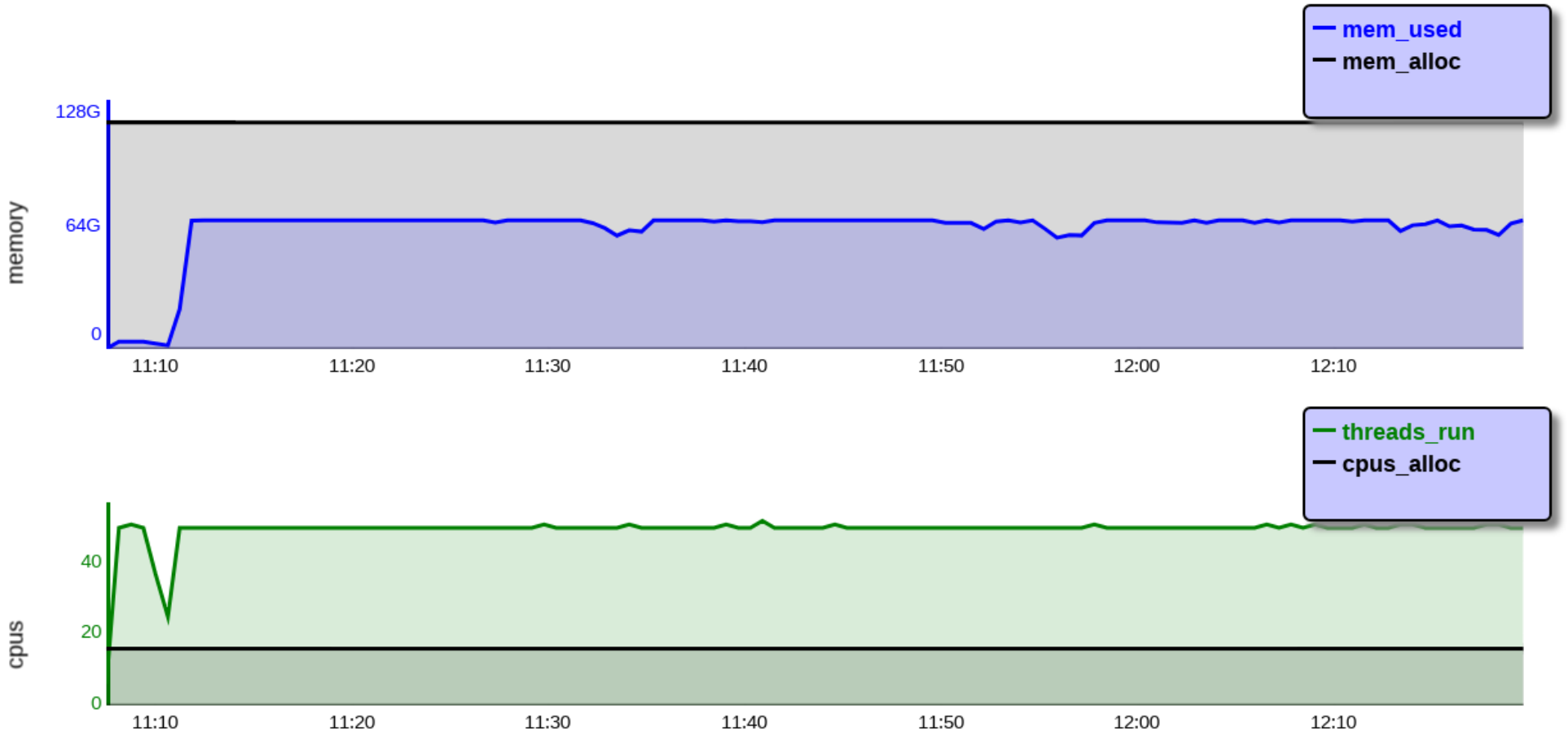
↓

Manta – Structural variants



Platinum genomes 30x coverage (germline)
*Bwa-mem (56 CPUs): 2.97 hours

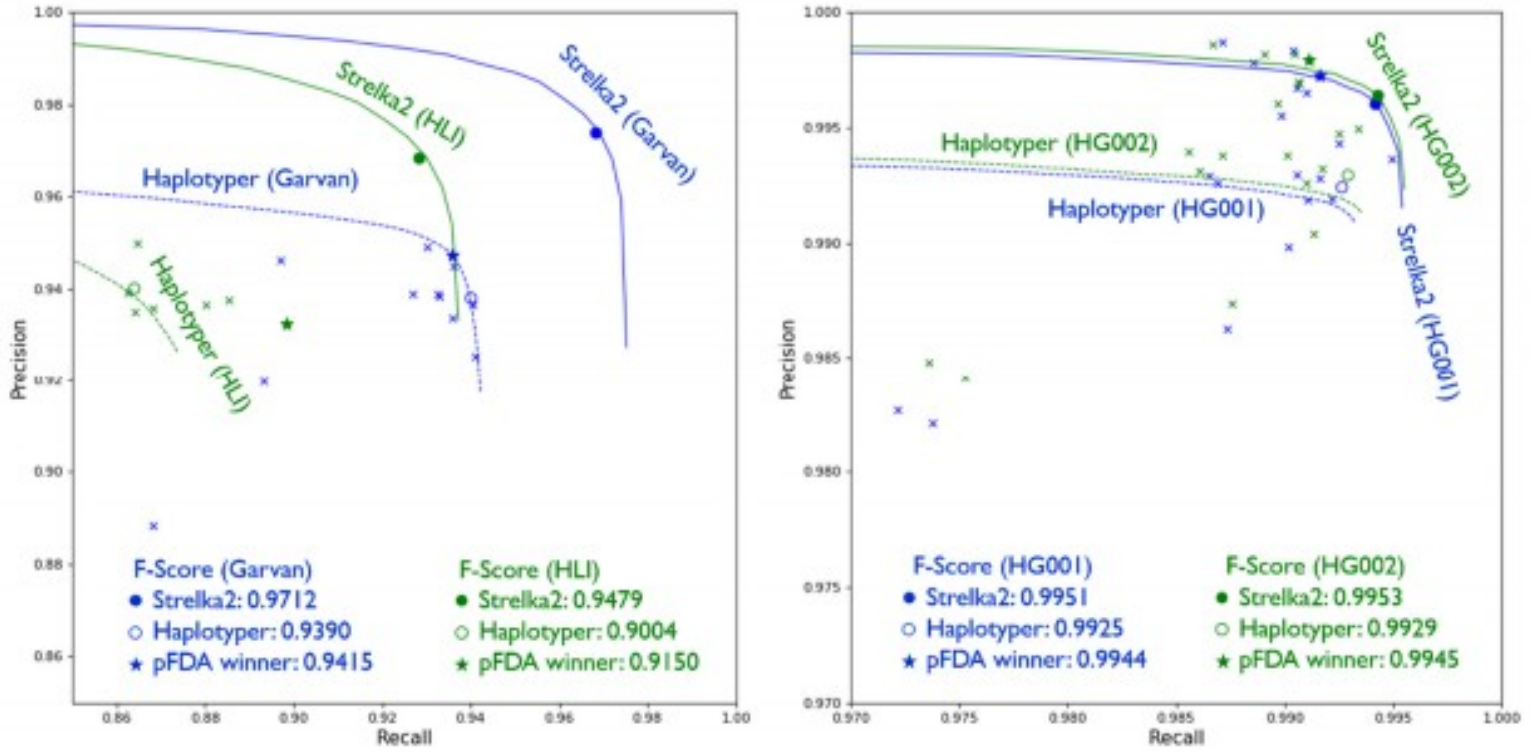
Strelka



Some applications overload the node. Must specify number of cpus as a parameter.

Strelka2: Fast and accurate variant calling for clinical sequencing applications

Sangtae Kim*[1], Konrad Scheffler*[1], Aaron L Halpern[1], Mitchell A Bekritsky[2], Eunho Noh[1], Morten Källberg[2,3], Xiaoyu Chen[1], Doruk Beyter[4], Peter Krusche[2], Christopher T Saunders[1]



Whole genome workflow in **< 5 hr**

- From fastq to VCFs and CNVs

Distributed Computing: Hail (Spark)

Node(s)	Time (sec)	Efficiency
1	1218.2	1.0
2	797.1	**0.76**
4	579.2	0.53
6	544.2	0.37
8	535.5	0.28

Time for converting genome vcf (platinum genomes) to parquet format

Deep learning: using GPUs

Not all software (apps) are not written for GPUs

- CUDA – parallel computing platform and programming model (NVIDIA proprietary)

- OpenCL – framework for writing programs across heterogenous platforms

- Digits

- Tensorflow

- Caffe2

Digits 6.0: Deep Learning GUI

DIGITS | New Model | teja_mnist (Logout) | Info | About

New Image Classification Model

Select Dataset

MNIST

MNIST

Done 02:45:54 PM

Image Size
28x28
Image Type
GRAYSCALE
DB backend
lmdb
Create DB (train)
45002 images
Create DB (val)
14998 images

Python Layers

Server-side file

Use client-side file

Solver Options

Training epochs
100

Snapshot interval (in epochs)
1

Validation interval (in epochs)
1

Random seed
[none]

Batch size multiples allowed
[network defaults]

Batch Accumulation

Solver type
SGD (Stochastic Gradient Descent)

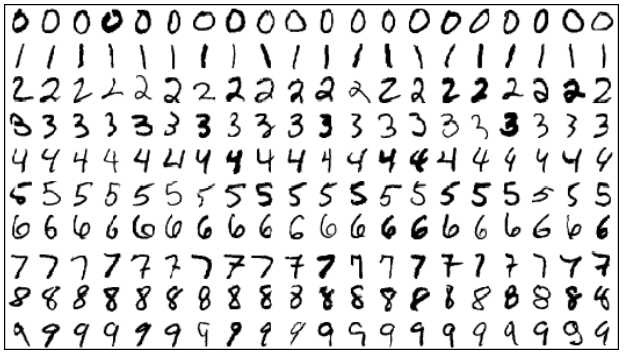
Base Learning Rate multiples allowed
0.01

Show advanced learning rate options

Data Transformations

Subtract Mean
Image

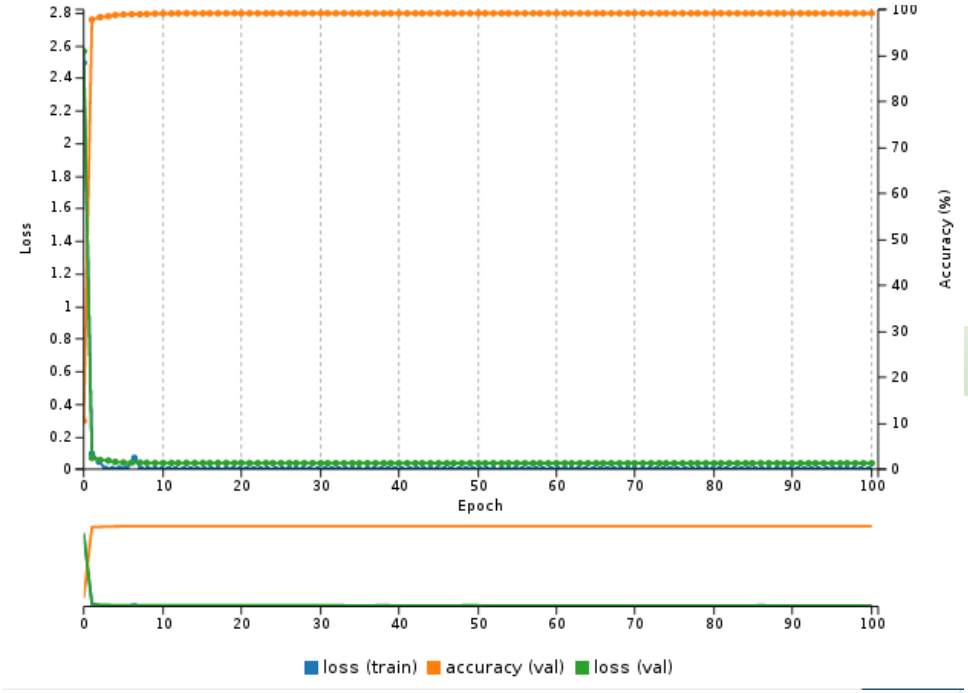
Crop Size
none



MNIST (handwritten digits)
LeNet Model

https://github.com/NVIDIA/DIGITS/blob/digits-6.0/docs/GettingStarted.md

Learning GUI



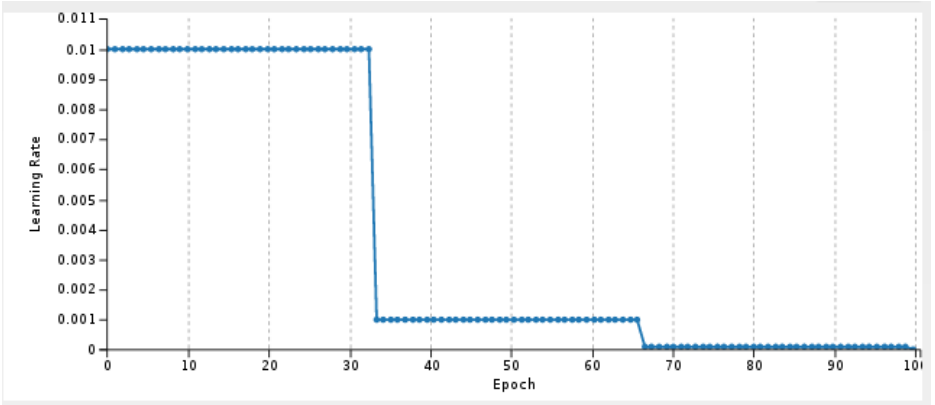
With 1 GPU

Job Status Done

- Initialized at 02:55:02 PM (1 second)
- Running at 02:55:03 PM (7 minutes, 39 seconds)
- Done at 03:02:42 PM

(Total - 7 minutes, 40 seconds)

Train Caffe Model Done



With 4 GPUs

Job Status Done

- Initialized at 04:12:04 PM (1 second)
- Running at 04:12:05 PM (6 minutes, 12 seconds)
- Done at 04:18:18 PM

(Total - 6 minutes, 13 seconds)

Train Caffe Model Done

Hardware

Tesla K80 (#0)

Memory
205 MB / 11.9 GB (1.7%)
GPU Utilization
85%
Temperature
68 °C

Tesla K80 (#1)

Memory
132 MB / 11.9 GB (1.1%)
GPU Utilization
86%
Temperature
52 °C

Tesla K80 (#2)

Memory
132 MB / 11.9 GB (1.1%)
GPU Utilization
85%
Temperature
63 °C

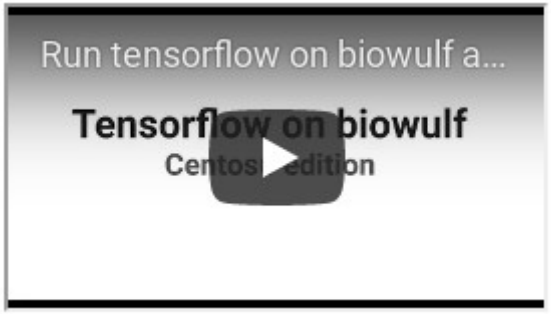
Tesla K80 (#3)

Memory
132 MB / 11.9 GB (1.1%)
GPU Utilization
85%
Temperature
50 °C

Process #18163

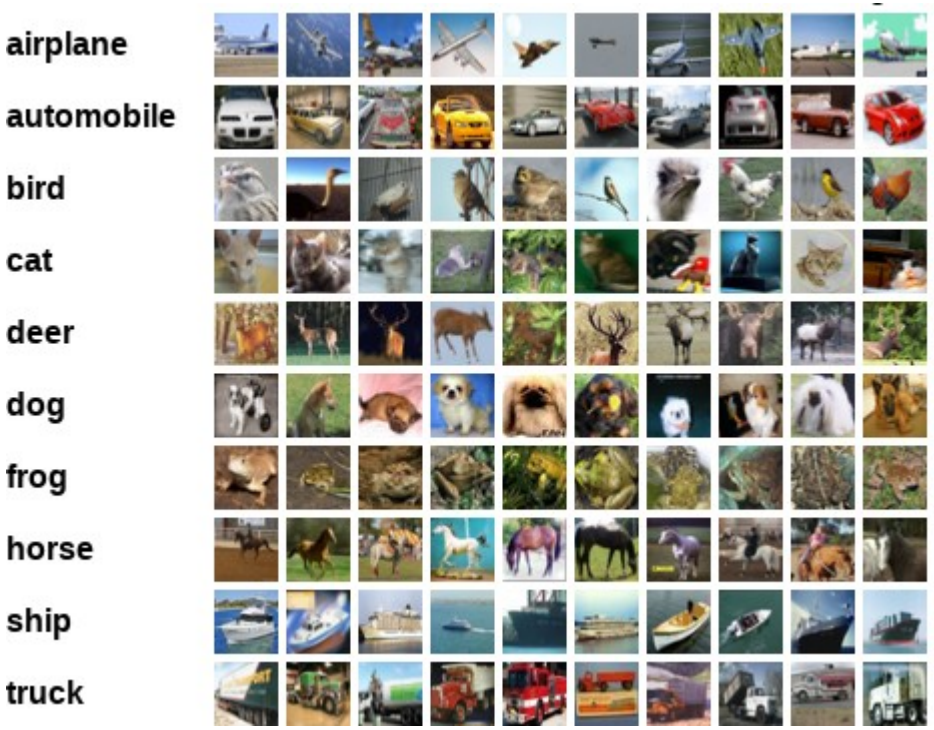
CPU Utilization
553.7%
Memory
1.29 GB (0.5%)

Tensorflow

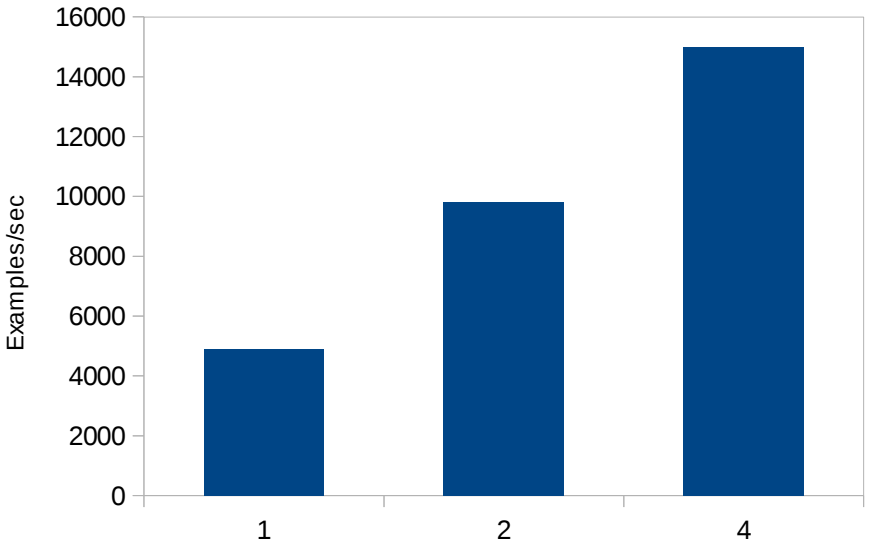


TensorFlow and Tensorboard
Train a DNN with tensorflow in a Singularity container on a GPU node and monitor the training progess with Tensorboard through an ssh tunnel from your local desktop (Mac/Linus and Windows).

https://hpc.nih.gov/docs/trainingvids.html

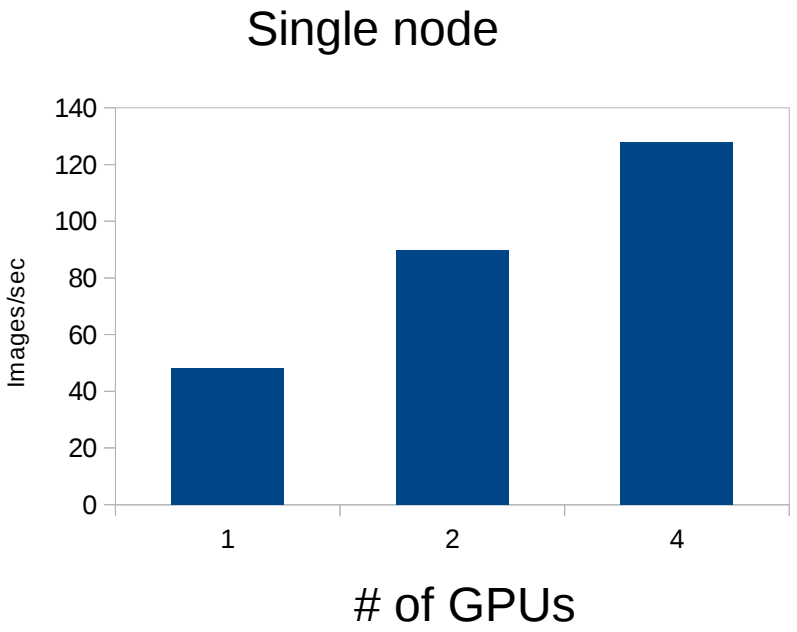


CIFAR 10 (60K 32x32 colored images)



- Small convolutional neural network (CIFAR)

Caffe2



- Training cars and boats (640 each) Imagenet
- Resnet50 (50 layers)
- Less efficient in multi-GPU multinode (with all 4 GPUs per node)
- Strategy: multi-node, 1 GPU per node

NIH HPC Facility Staff


Steve Bailey


Steven Fellini, Ph.D.


Susan Chacko, Ph.D.


Gennady Denisov, Ph.D.

Picture unavailable

Afif Elghraoui


Ainsley A. Gibson, Ph.D.


David Hoover, Ph.D.


Patsy Jones

Picture unavailable

Charles Lehr


Jean Mao, Ph.D.


Tim Miller


Charlene Osborn


Mark Patkus


Dan Reisman


Wolfgang Resch, Ph.D.


Jerez Te, Ph.D.


Rick Troxel


Sylvia Wilkerson


Michael Harris (intern)

Questions: staff@hpc.nih.gov