~~Protein~~Biomolecular structure prediction with AI Alphafold and friends

Martin Čuma
Center for High Performance Computing
University of Utah
m.cuma@utah.edu

Outline

- Why AI structure prediction
- How it works?
- Some history (incl. Nobel Prize)
- Performance considerations
- Tools we have available and their efficient use

Biomolecular structure prediction

Protein structure

- Proteins are one of the base building blocks of life
- They form 3D structure which is affected by the amino acid sequence and surroundings
- Protein structure prediction
  – experimental - X-ray crystallography, NMR spectroscopy, Cryo electron microscopy - labor intensive, expensive
  – computational
    - comparative - assembly from known smaller structures
    - ab-initio - physical principles structure minimization (molecular dynamics, ...)

AI structure prediction

- requires Multiple Sequence Aligned (MSA) input
  - identifies relationships between sequences
  - first step, usually runs only on CPUs and uses large databases (is I/O intensive)
- use deep neural network trained on known (protein) structures
  - runs on GPUs, or CPUs (very slowly)
  - larger sequences need GPUs with more memory
- AI predicted structure may follow up with MD minimization

AI structure prediction

- AI revolutionized structure prediction
  - much better accuracy than previous methods (70% for Alphafold2)
  - easier to use and quicker results
- 2024 Nobel Prize for Chemistry
  - D. Baker (U Wash) - Rosetta(Fold) - "for computational protein design"
  - D. Hassabis, J. Jumper (Google DeepMind) - AlphaFold - "for protein structure prediction"

AI structure prediction programs available

SP modules (easy)

- Alphafold 2, 3, https://github.com/google-deepmind/alphafold3
  - be careful about the MSA performance
- Colabfold(batch), https://github.com/sokrypton/ColabFold
  - more efficient MSA, Alphafold for the inference
- Boltz1, https://github.com/jwohlwend/boltz
  - fully open source AF alternative, uses Colabfold server for MSA

SP other tools (advanced)

- RFAntibody, https://github.com/RosettaCommons/RFantibody
  - workflow must be run in a container
- Google Colab notebooks
  - RFDiffusion, Alphafold, ProteinMPNN
  - can run on CHPC resources, more difficult setup
- RFDiffusion, https://github.com/RosettaCommons/RFdiffusion
  - trickier to set up due to user space requirements
- RosettaFold All-Atom, https://github.com/baker-laboratory/RoseTTAFold-All-Atom
  - open source alternative to Alphafold 3, requires setup in user space

Performance considerations

Performance considerations

- AI enabled prediction runs in 2-3 steps
- Step 1 - MSA
  - Mostly on CPU (except for mmseqs-gpu), very I/O intensive
  - it's worth to have the databases or at least their indices in RAM
- Step 2 - AI structure inference
  - Much more efficient on GPUs
  - Can be faster than the MSA search
- Step 3 (optional, AF2) - MD structure refinement
  - GPU or CPU, GPU faster for larger structures

MSA

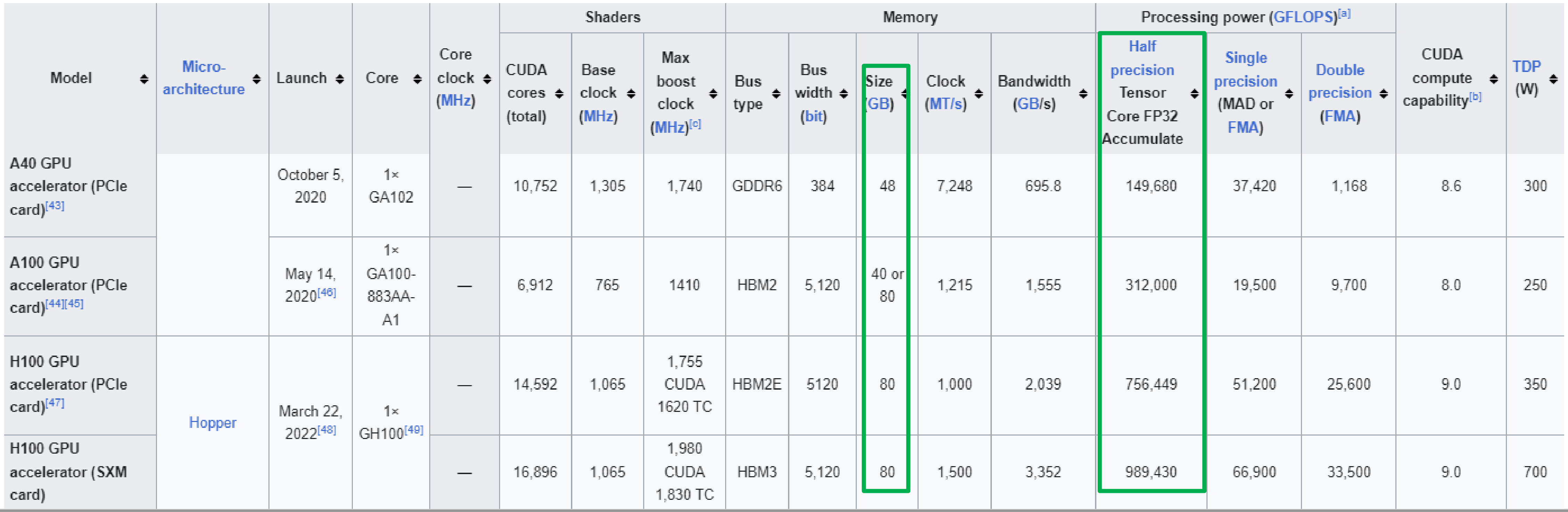
- Alphafold uses more accurate but much slower MSA program
  - AF2 uses indexed databases - indices in the RAM disk (~30 GB)
  - AF3 does not = all databases on VAST file system
  - ~ 30 min for reference protein 779 b.p., AF3 a bit slower
  - for that reason 2 jobs, one CPU job for MSA, one GPU job for inference
- Colabfold, Boltz
  - use MSA server which runs mmseqs2
  - CHPC server buffers the important databases in RAM (700 GB)
  - server stores past MSAs so it can reuse them and return result quicker
  - ~ 5 min for reference protein
  - just a single job on a GPU is usually OK

GPU types

- CHPC has GPUs from many different generations
  - Their performance and capabilities vary widely
  - https://www.chpc.utah.edu/documentation/guides/gpus-accelerators.php#gpu_types
- (Nvidia) GPU classification:
  - Generation (code name - Maxwell, Pascal, Volta, Turing, Ampere, Hopper)
  - Compute Capability (5.2, 6.0, 6.1, 7.0, 7.5, 8.0, 8.6, 8.9, 9.0)
    https://developer.nvidia.com/cuda-gpus#compute
  - Theoretical compute throughput (single, double precision, tensor)
    https://en.wikipedia.org/wiki/Nvidia_Tesla
    https://en.wikipedia.org/wiki/Quadro
    https://en.wikipedia.org/wiki/GeForce
    https://en.wikipedia.org/wiki/GeForce_40_series
  - Amount of memory (~10-80 GB)

Data center GPUs

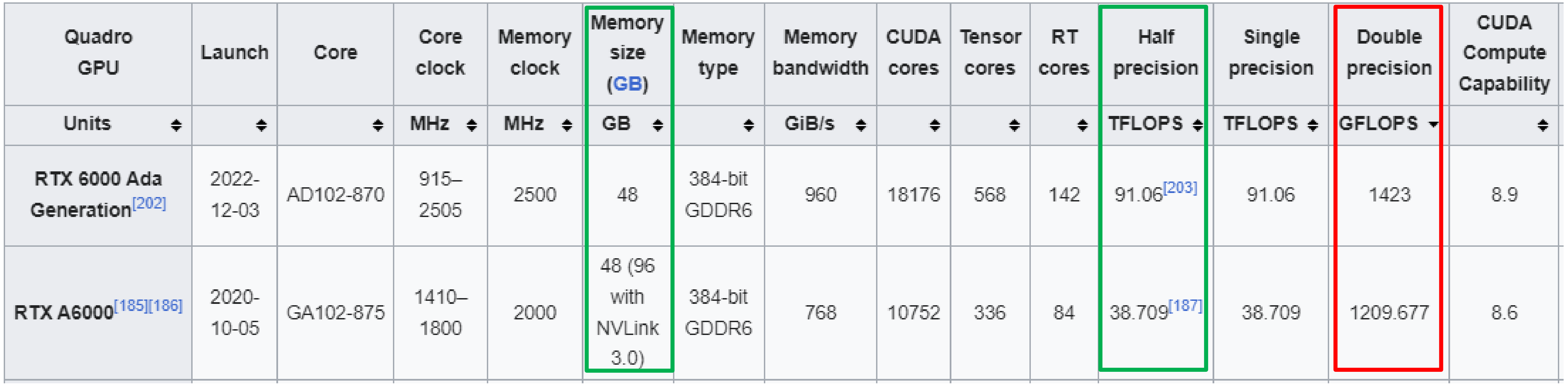
- Example: A40, A100, H100
  - Data center GPUs, high double precision performance, large memory
  - Expensive (H100 ~$20000 w/ edu discount), hard to get
  - Good for simulations that need high numerical precision (engineering), or high memory (AI)



Model	Micro-architecture	Launch	Core	Core clock (MHz)	Shaders			Memory					Processing power (GFLOPS)[a]			CUDA compute capability[b]	TDP (W)
					CUDA cores (total)	Base clock (MHz)	Max boost clock (MHz)[c]	Bus type	Bus width (bit)	Size (GB)	Clock (MT/s)	Bandwidth (GB/s)	Half precision Tensor Core FP32 Accumulate	Single precision (MAD or FMA)	Double precision (FMA)		
A40 GPU accelerator (PCIe card)[43]		October 5, 2020	1× GA102	—	10,752	1,305	1,740	GDDR6	384	48	7,248	695.8	149,680	37,420	1,168	8.6	300
A100 GPU accelerator (PCIe card)[44][45]		May 14, 2020[46]	1× GA100-883AA-A1	—	6,912	765	1410	HBM2	5,120	40 or 80	1,215	1,555	312,000	19,500	9,700	8.0	250
H100 GPU accelerator (PCIe card)[47]	Hopper	March 22, 2022[48]	1× GH100[49]	—	14,592	1,065	1,755 CUDA 1620 TC	HBM2E	5120	80	1,000	2,039	756,449	51,200	25,600	9.0	350
H100 GPU accelerator (SXM card)				—	16,896	1,065	1,980 CUDA 1,830 TC	HBM3	5,120	80	1,500	3,352	989,430	66,900	33,500	9.0	700

Professional GPUs

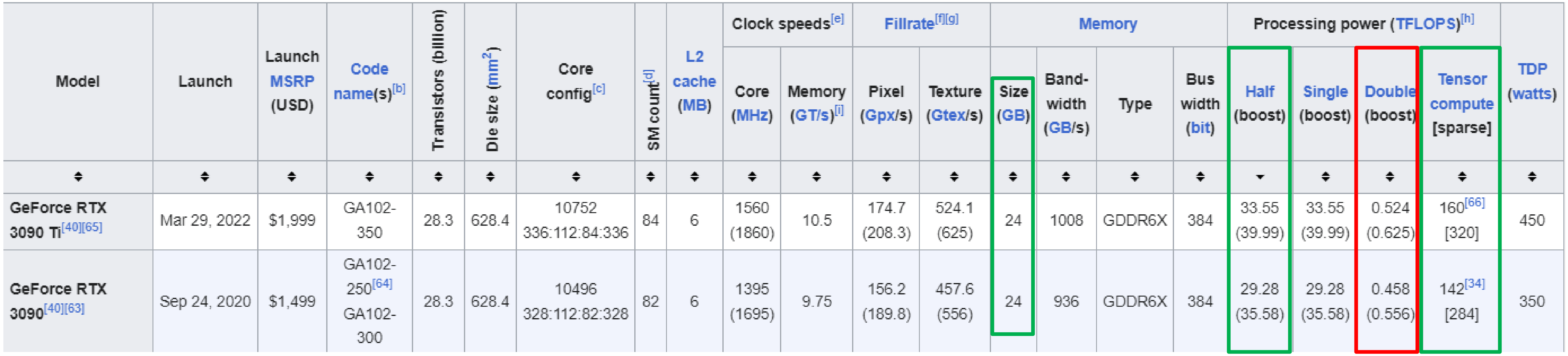
- Example: RTX6000, A6000
  - Mid size models (graphical workstations), more memory than the gaming graphics cards, price about 1/2 to 1/3 of the data center GPUs
  - Very low double precision performance, good single and tensor performance
  - Good for lower precision numerical calculations, AI with small to medium size models



Quadro GPU	Launch	Core	Core clock	Memory clock	Memory size (GB)	Memory type	Memory bandwidth	CUDA cores	Tensor cores	RT cores	Half precision	Single precision	Double precision	CUDA Compute Capability
Units			MHz	MHz	GB		GiB/s				TFLOPS	TFLOPS	GFLOPS	
RTX 6000 Ada Generation[202]	2022-12-03	AD102-870	915–2505	2500	48	384-bit GDDR6	960	18176	568	142	91.06[203]	91.06	1423	8.9
RTX A6000[185][186]	2020-10-05	GA102-875	1410–1800	2000	48 (96 with NVLink 3.0)	384-bit GDDR6	768	10752	336	84	38.709[187]	38.709	1209.677	8.6

Consumer GPUS

- Example: RTX3090
  - Consumer grade graphics cards, recent don't fit to compute servers
  - Very low double precision performance, good single and tensor performance
  - More affordable (a few thousand $)
  - Good for lower precision numerical calculations, AI with small to medium size models



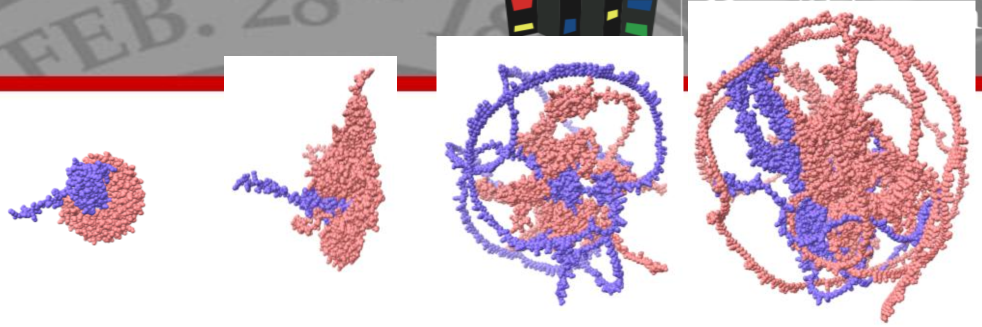
Model	Launch	Launch MSRP (USD)	Code name(s)[b]	Transistors (billion)	Die size (mm^2)	Core config[c]	SM count[d]	L2 cache (MB)	Clock speeds[e]		Fillrate[f][g]		Memory				Processing power (TFLOPS)[h]				TDP (watts)
									Core (MHz)	Memory (GT/s)[i]	Pixel (Gpx/s)	Texture (Gtex/s)	Size (GB)	Band-width (GB/s)	Type	Bus width (bit)	Half (boost)	Single (boost)	Double (boost)	Tensor compute [sparse]	
GeForce RTX 3090 Ti[40][65]	Mar 29, 2022	$1,999	GA102-350	28.3	628.4	10752 336:112:84:336	84	6	1560 (1860)	10.5	174.7 (208.3)	524.1 (625)	24	1008	GDDR6X	384	33.55 (39.99)	33.55 (39.99)	0.524 (0.625)	160[66] [320]	450
GeForce RTX 3090[40][63]	Sep 24, 2020	$1,499	GA102-250[64] GA102-300	28.3	628.4	10496 328:112:82:328	82	6	1395 (1695)	9.75	156.2 (189.8)	457.6 (556)	24	936	GDDR6X	384	29.28 (35.58)	29.28 (35.58)	0.458 (0.556)	142[34] [284]	350

How long does it take?

Variables

- Protein size
- GPU type
- GPU availability
- Job parallelization

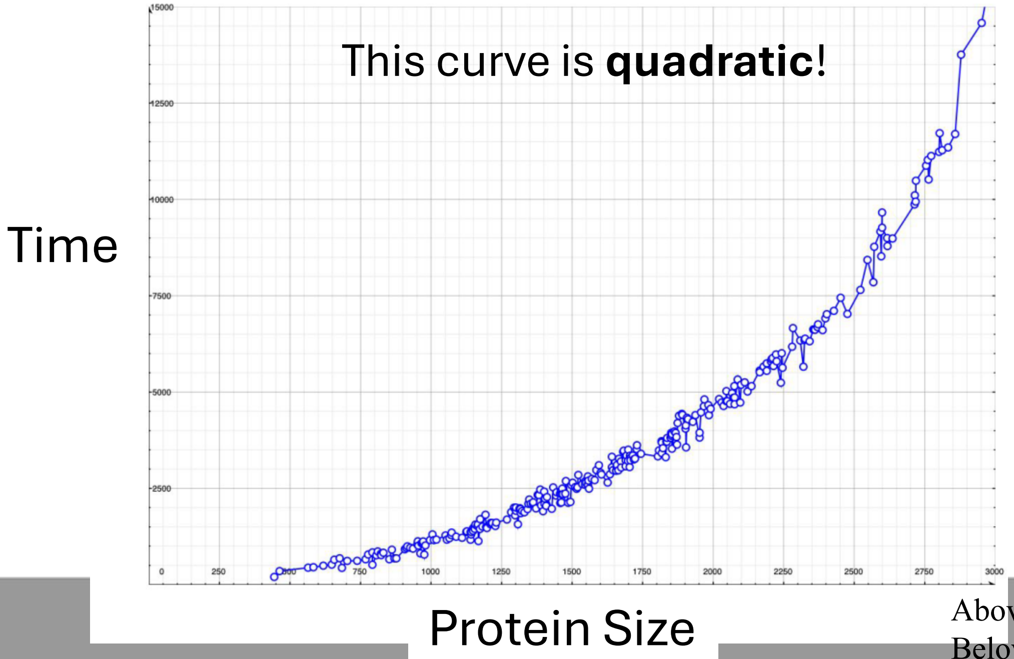
How long does it take?



Protein 1	mRnase1	Insulin	Bnl	Dl
Protein 2	mRnh1	InsRECD	Btl	N
# Amino acids	606aa	1040aa	1829aa	3537aa
Runtime (3090 GPU)	7 min	24 min	75 min	828 min

Variables

- **Protein size**
- GPU type
- GPU availability
- Job parallelization



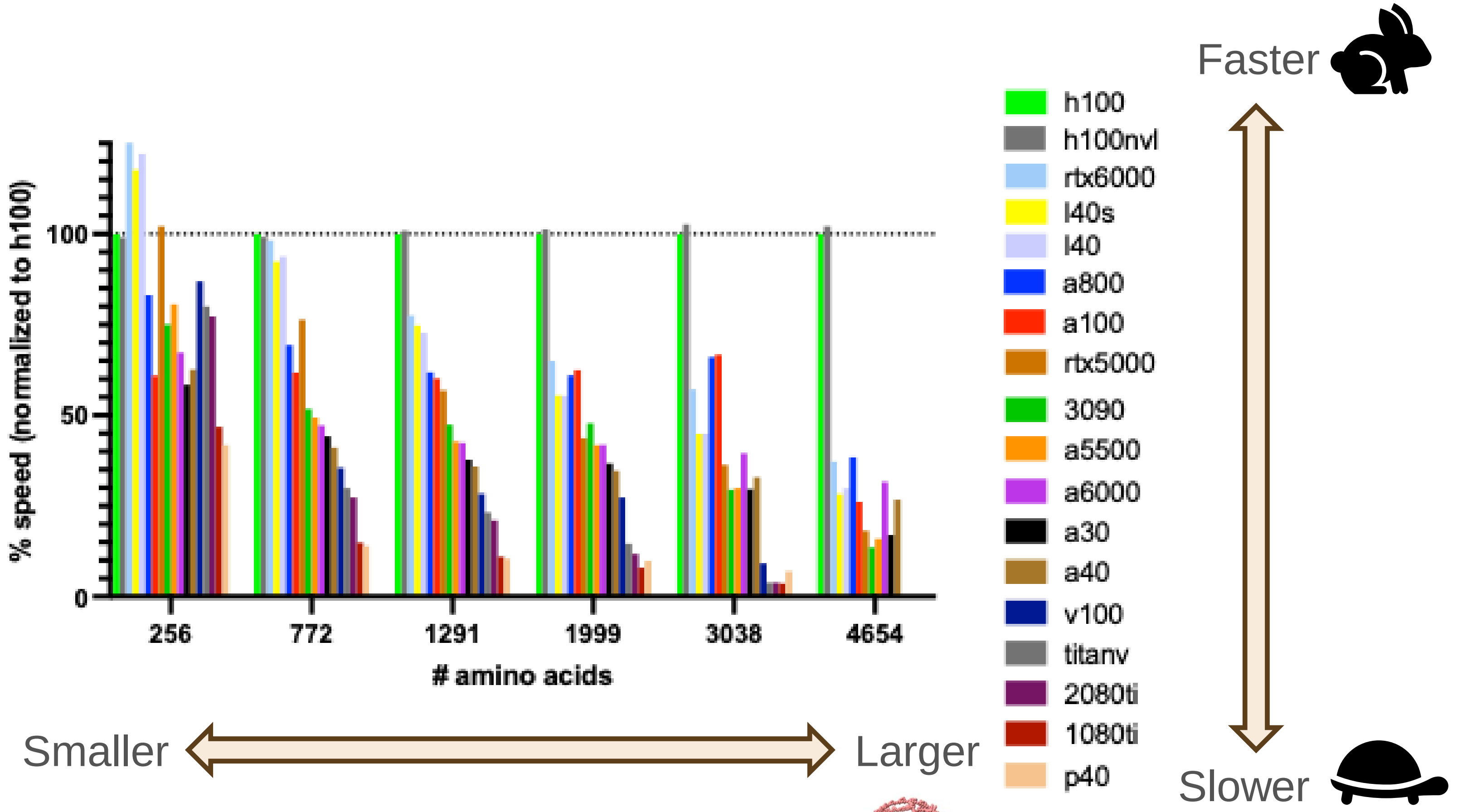
Above: Justin Bosch, UofU
Below: Tom Goddard, UCSF

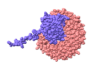
How long does it take?

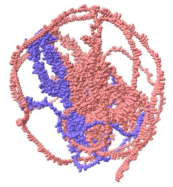
Variables

- Protein size
- **GPU type**
- GPU availability
- Job parallelization

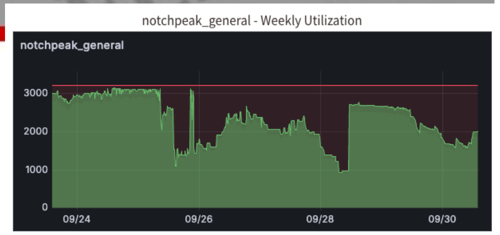
Just the GPU run inference, not the MSA







How long does it take?



Variables

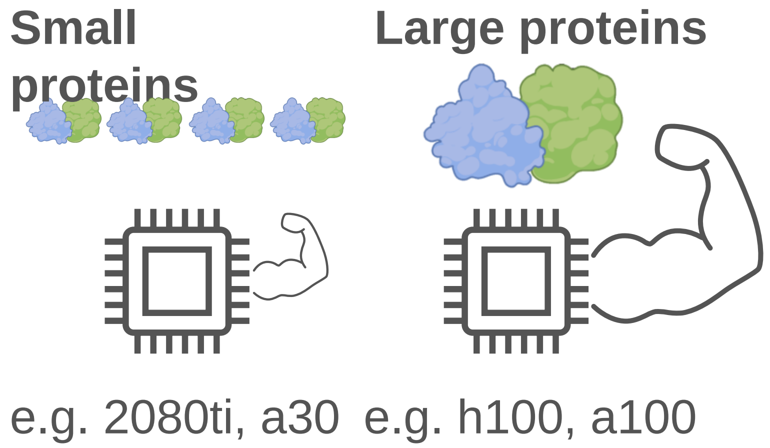
- Protein size
- GPU type
- **GPU availability**
- Job parallelization

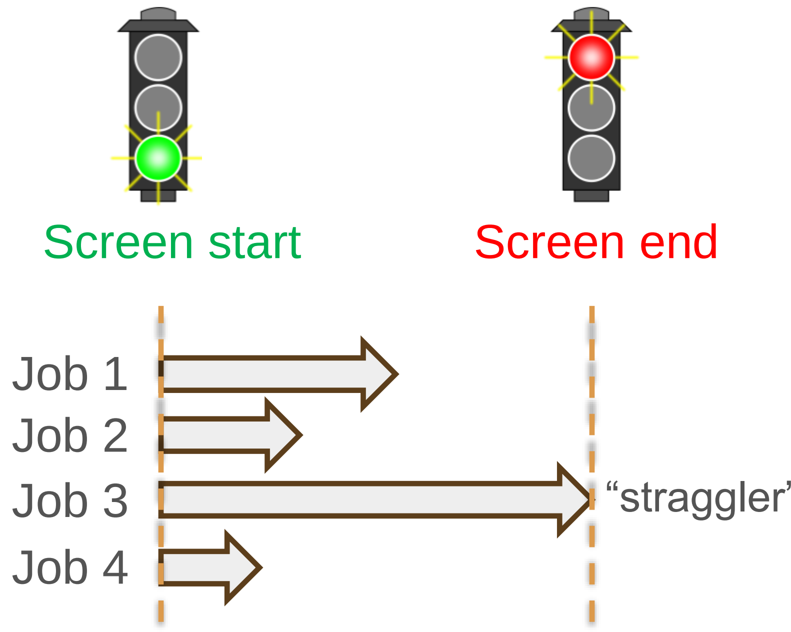
	Notchpeak		Lonepeak	Kingspeak		Redwood		Granite				
GPU	CHPC	Owner	CHPC	CHPC	Owner	CHPC	Owner	CHPC	Owner	CHPC Sum	Owner Sum	Total Sum
h100nvl	0	4	0	0	0	0	0	4	0	4	4	8
h100	0	0	0	0	0	0	1	0	0	0	1	1
rtx6000	0	9	0	0	0	0	1	0	4	0	14	14
l40s	0	0	0	0	0	0	0	16	0	16	0	16
l40	0	10	0	0	0	0	0	0	0	0	10	10
a100	4	23	0	0	0	0	21	0	0	4	44	48
a800	0	9	0	0	0	0	0	0	0	0	9	9
rtx5000	0	0	0	0	0	0	8	0	1	0	9	9
3090	12	9	0	0	0	0	0	0	0	12	9	21
a6000	0	25	0	0	0	0	0	0	0	0	25	25
a5500	0	6	0	0	0	0	0	0	0	0	6	6
rtx2000	0	0	0	0	0	0	0	0	3	0	3	3
a40	0	30	0	0	0	0	4	0	0	0	34	34
a30	0	0	0	0	0	0	20	0	0	0	20	20
v100	9	2	0	0	0	0	0	0	0	9	2	11
titanv	0	4	0	0	0	0	0	0	0	0	4	4
2080ti	22	53	0	0	0	0	0	0	0	22	53	75
1080ti	0	8	158	0	0	53	0	0	0	211	8	219
p40	1	0	0	0	0	0	0	0	0	1	0	1
Sum	48	192	158	0	0	53	55	20	8	279	255	534

How long does it take?

Variables

- Protein size
- GPU type
- GPU availability
- **Job parallelization**





Info on how to get what GPUs:
https://www.chpc.utah.edu/presentations/images-and-pdfs/usinggpus24f.pdf

Use of Biomolecular structure prediction programs (easy)

Alphafold 2

- Installed years ago when had JBOD NFS file servers
  - very slow MSA search (15 hrs reference sequence 779 bp)
  - only marginally better with spinner local disk (10 hrs)
  - copy database indices into RAM disk (~20-30GB), ~10x speedup over NFS file server, ~2x over VAST (1:15 hr)
  - custom separate MSA and inference into 2 jobs, MSA on CPUs, inference on GPUs
    - 2nd job automatically submitted by the first job, 2nd job waits till 1st job finishes
    - 1st job uses CPUs and more RAM (DBs on RAM disk)
    - 2nd job uses GPU, no need for DBs

Alphafold 2

- Installed in a container
- alias run_alphafold.sh command with paths to the databases via Lmod module
- 1st job SLURM script - `sbatch run_alphafold_chpc_232.slr`

```
ml alphafold/2.3.2
export FASTA_FILE="t1050.fasta"
export OUTPUT_DIR="out"
/uufs/chpc.utah.edu/sys/installdir/alphafold/db_to_tmp_232.sh
SCRDB=/scratch/general/vast/app-repo/alphafold
TMPDB=/tmp/$SLURM_JOBID
sbatch -d afterok:$SLURM_JOBID run_alphafold_chpc_232_2.slr
run_alphafold_full.sh --use_gpu_relax --fasta_paths=$FASTA_FILE --output_dir=$OUTPUT_DIR --
max_template_date=2022-01-01 --run_feature=1
```

Copy MSA db indices to RAM disk

Submit GPU job for inference

Custom option to only do MSA

- Detailed instructions
  - https://www.chpc.utah.edu/documentation/software/alphafold.php#alphafold

Alphafold 3

- Databases are not indexed - all on VAST
  - – a bit slower than indices in RAM, but need less RAM
- AF3 has option to split the MSA and inference - use that
- Parameters license restriction
  - – require users to request access to the parameters,
  - – e-mail approval to helpdesk@chpc.utah.edu
  - – we add to "alphafold3" group which has access to the centrally installed parameters

Alphafold 3

- Installed in a container
- 2 jobs, like in Alphafold 2
- Not just proteins, uses json file for input

```
ml alphafold/3.0.1
export INPUT_FILE="af_input.json"
export OUTPUT_DIR="out"
sbatch -d afterok:$SLURM_JOBID run_alphafold_chpc_301_2.slr
run_alphafold.sh --json_path=$INPUT_FILE --output_dir=$OUTPUT_DIR --norun_inference
```

- Inference

```
export INPUT_FILE="out/2pv7/2pv7_data.json"
run_alphafold.sh --json_path=$INPUT_FILE --output_dir=$OUTPUT_DIR --norun_data_pipeline
# to run on older GPUs, add --flash_attention_implementation=xla
```

- Detailed instructions
  - https://www.chpc.utah.edu/documentation/software/alphafold.php#alphafold3

ColabFold / LocalColabfold

- Discovered when trying to find faster MSA alternative
  - different, faster MSA program (mmseqs2)
- MSA search done at remote ColabFold server by default
  - we run our own ColabFold (mmseqs2) server

```
colabfold_batch --host-url=http://colabfold01.int.chpc.utah.edu:8088
```

- use LocalColabFold for making Miniforge environment
  - Generally faster than Alphafold (because of the faster MSA)
  - Uses Alphafold 2 for inference
- https://www.chpc.utah.edu/documentation/software/alphafold.php#colabfold

ColabFold API server

- One researcher got banned from ColabFold server and asked us to set one up locally
- For good performance buffer some databases in RAM (~0.7 TB) and the rest on SSD drive (~1 TB)
- We found 10yo donated 32 core server with 1 TB RAM and 1 TB SSD
  - copy the non-RAM databases to local SSD, RAM databases on network file server symlinked to local SSD
  - "jobs" directory that caches past jobs on network file server

Boltz-1

- Boltz - open source alternative to Alphafold 3
- Easy installation into venv with pip
- Uses mmseqs2 for MSA search, like ColabFold
- Can use the ColabFold server

```
ml boltz1/0.4.1
boltz predict $FASTA_FILE --use_msa_server
--msa_server_url=http://colabfold01.int.chpc.utah.edu:8088
```

- Single GPU job, fasta or yaml file for input
- Can work with multiple biomolecule types
- Detailed instructions
  - https://www.chpc.utah.edu/documentation/software/alphafold.php#boltz

Use of Biomolecular structure prediction programs (difficult)

Google Colab

- Alphafold, RFdiffusion, ProteinMPNN in Colab notebook
  - useful esp. for the Baker lab tools which often require user space installation
  - https://colab.research.google.com/github/sokrypton/ColabFold/blob/main/AlphaFold2.ipynb
  - https://colab.research.google.com/github/sokrypton/ColabDesign/blob/main/rf/examples/diffusion.ipynb
- User requested to run Colab on their owner node
- We have Colab container for local runtime
- https://www.chpc.utah.edu/documentation/software/google-colab.php

Google Colab notebooks

- Free Colab has 15 min execution limit
- To run at CHPC with Jupyter notebook on local browser
  - start Jupyter with Colab environment as a SLURM job
  - create SSH tunnel to this job from local laptop/desktop
  - after opening Colab notebook choose the "local runtime"
  - run the notebook
- Detailed instructions
  - https://www.chpc.utah.edu/documentation/software/google-colab.php

Google Colab notebooks

- Most notebooks require local installation of the dependencies
  - install inside of the Colab notebook or manually in cluster terminal
  - may need to change hard coded paths in the notebook (e.g. /usr/local) to paths in user's CHPC home directory
  - not for the faint hearted but for the most part doable
- UI issues (minor inconveniences)
  - the Colab notebook resource monitor shows the whole node, not its part allocated to the job
  - GPU and disk resource monitor are not correct
  - the Colab notebook's file manager does not work - use one that comes with Jupyter

RFAntibody

- More like a workflow
- User installable, and requires newer OS
  - typical example of packages coming from RosettaCommons/Baker Lab
  - generally require some kind of writeability to where the programs are installed - won't work for our typical centralized installations
  - workaround in this case is to set it dependencies in a container, bind-mount current directory to /home in the container and install/run the package in the /home
- In the next slide we show installation instructions modified for running in an Apptainer container

RFAntibody

- Shell into a container that has RFAntibody base dependencies

```
cd <directory where you want to install the program>
ml apptainer
apptainer shell --nv -B .:/home
/uufs/chpc.utah.edu/common/home/u0101881/containers/singularity/containers/rfantibod
y/rfa.sif
```

- In the container, get a RFantibody fork with fixed examples and download the weights

```
git clone https://github.com/katyachemistry/RFantibody.git
cd RFantibody
git checkout fixed
bash ./include/download_weights.sh
```

- For the dependencies setup, need to run "poetry install" outside of the bash script, otherwise it creates new venv.

```
nano include/setup.sh
```

comment out

```
#poetry install &&
```

RFAntibody

- then run the dependency installation:

```
bash include/setup.sh
poetry install
```

- The program is installed, now you can run the example

```
bash /home/scripts/examples/rfdiffusion/antibody_pdbdesign.sh
```

This needs to be run on a GPU which a decent size memory, it ran out of GPU memory on my desktop which only has 2 GB GPU.

- If you need to run other simulation, you don't need to repeat the steps above, just shell into the container in the same directory you have the program installed

```
apptainer shell --nv -B .:/home
/uufs/chpc.utah.edu/common/home/u0101881/containers/singularity/containers/rfantibod
y/rfa.sif
```

RFdiffusion, RosettaFold

- Install in the works, or install yourself
- RFdiffusion is designed for user space - have to modify for shared environment
- RosettaFold All Atom - similar functionality to Alphafold 3 or Boltz
- Overall issue with tools from RosettaCommons is that they are designed to be installed by user, which requires modifications if it's installed by us for everyone
- Therefore it may be easier for each user to install it themselves
- For that reason we're not big fans of these tools

Questions?

Survey:

https://tinyurl.com/yt4fvauk