National Supercomputing Centre

Applying for HPC Resources: A Planning Workshop

About this document

- **Target Audience:** NSCC Project Applicants, including Principal Investigators (PI) and personnel authorised by the PI.
- **Purpose:** Provide a structured, simple method to estimate the resources needed for your project, supporting cost control and project feasibility.

The Basics – What are we counting?

To apply for resources, you need to understand two main units of measurement:

- **CPU Core-Hours:** One CPU core being used for an hour.
  - *Example:* 256 CPU cores used for 1 hour = 256 CPU core-hours. 128 CPU cores used for 2 hours = 256 CPU core-hours.
- **GPU Card-Hours:** One GPU card being used or reserved for an hour.
  - *Example:* 4 GPUs used for 4 hours = 16 GPU card-hours.
  - CPU cores are bundled together with the GPU cards requested. There's no need to request for additional CPU core-hours when requesting for GPU card-hours.
  - *Example:* 1,000 GPU card-hours are requested. There's no need to put it an additional request of 16,000 CPU core-hours on ASPIRE 2A.

The HPC Systems (ASPIRE 2A, ASPIRE 2A+, & Upcoming System)

- **System Assignment:** Applicants do not choose among ASPIRE 2A, ASPIRE 2A+, and upcoming system; NSCC assigns the most suitable system based on your project requirements.
- **Shared System Rule:** HPC systems are shared environments, not dedicated cloud rentals.
  - **Do not** estimate resources assuming you will have exclusive, continuous access to hardware (e.g., reserving 256 GPUs at once).
  - Assuming exclusive access often leads to severe over-estimation. Only estimate the actual hours your specific jobs need to run.

The HPC Systems (ASPIRE 2A & ASPIRE 2A+)

ASPIRE 2A

Full list of HPC cluster node types and specification:

Server	CPU Model	Cores per socket	Socket per server	Total Physical cores per server	Available RAM (DDR4)	GPUs
Standard Compute Node (768 nodes)	Dual-CPU AMD EPYC 7713	64	2	128	512 GB	No GPU
GPU compute node (64 nodes)	Single-CPU AMD EPYC 7713	64	1	64	512 GB	**4x Nvidia A100 40GB**
GPU AI Node (12 nodes)	Single-CPU AMD EPYC 7713	64	1	64	512 GB	**4xNvidia A100 40GB** (11TB nvme)
GPU AI Node (6 nodes)	Dual-CPU AMD EPYC 7713	64	2	128	1 TB	**8xNvidia A100 40GB** (14TB nvme)
Large memory node (12 nodes)	Dual-CPU AMD EPYC 7713	64	2	128	2 TB	No GPU
Large memory node (4 nodes)	Dual-CPU AMD EPYC 7713	64	2	128	4 TB	No GPU

Interconnect (Slingshot 10)

1x100G Link (CPU node, Large Memory Node) ; 2x100G Link (GPU Node)

Storage

Home+Project (GPFS) ≈ 25 PB ; Scratch (Lustre) ≈ 10 PB

ASPIRE 2A+

ASPIRE2A+ Components	Specification
Compute Nodes	NVIDIA DGX SuperPOD™ - 40 nodes of DGX H100 8 x NVIDIA H100 80GB per node
Interconnect	NVIDIA Quantum 2 based NDR InfiniBand
Storage	Home ~ 27.5 PB, Scratch ~ 2.5 PB

The HPC Systems (Upcoming System)

Upcoming System

Server	CPU Model	Cores Per Socket	Socket Per Server	Total Physical Cores per Server	Availabe RAM (DDR5)	GPU	Local Disk	Number of Slingshot Card
CPU Node (768x Node)	Dual-Socket AMD EPYC 9655	96	2	192	768GB	N/A	N/A	1
GPU Node (192x Node)	Dual-Socket AMD EPYC 9655	96	2	192	2.25TB	8x NVIDIA H200 GPU 141 GB	15.36 TB NVMe	4
Large Memory Node (16x Node)	Dual-Socket AMD EPYC 9554	64	2	128	4TB	N/A	30.72 TB NVMe	1

Interconnect (Slingshot 12)

400 Gbps

Storage

Scratch (Lustre) ≈ 24 PB; Home (Lustre) ≈ 2.5 PB; Project (Lustre) ≈ 37 PB

The Golden Rule: Test, Multiply, Buffer

Instead of guessing or doing complex math, base your estimates on past workloads. Use this simple 3-step method:

1. **Test:** Run representative small benchmarks across multiple configurations. Look at your output file to see how long it took and how many cores/GPUs you used. Select the best configuration for your production runs.
2. **Multiply:** Multiply the best test run by the total amount of work you plan to do this year.
3. **Buffer:** Always include a 10% resource buffer for debugging, restarts, and unforeseen issues.

Tip: If available, use your past usage data to estimate how much resources you'll need to apply for.

No data? Don't have reference data for past workloads? Approach us at projects-admin@nscc.sg to request for a small resource allocation to establish a reference.

Example 1 - CPU workload

Scenario: Running 100 molecular dynamics simulations for a 1-year project.

- **Step 1 (Test):** We ran one test simulation. For the most optimal configuration, it took 38 hours using 2048 CPU cores. This equals 77,716 CPU core-hours (which we rounded up to 80,000 for safety) .
- **Step 2 (Multiply):** 80,000 CPU core-hours x 100 runs = 8,000,000 CPU core-hours.
- **Step 3 (Buffer):** Add a 10% buffer to the base amount.
- **Final Request:** 8,800,000 CPU Core-hours.

Example 2 - AI workload

Scenario: Training a Large Language Model.

- **Step 1 (Test):** Based on past workloads or benchmarks, we know training this model will take roughly 3 days on 16 A100 GPUs, consuming 1,152 A100 card hours.
- **Step 2 (Multiply):** 1,152 A100 GPU hours x 10 runs = 11,520 A100 GPU card-hours.
- **Step 3 (Buffer):** Add a 10% buffer.
- **Final GPU Request:** 12,672 A100 GPU Card-hours.

Don't Forget the CPU: Data pre/post-processing (like data cleaning and tokenization) runs mainly on CPUs, and should be run on CPU nodes where available. For example, we estimate 100,000 CPU core-hours, plus a 10% buffer, totaling 110,000 CPU Core-hours.

Estimating your storage needs

Apply the same 3-step method to your storage. If your test run generates 3 GB of data, and you run it 100 times, that is 300 GB, plus a 10% buffer = 330 GB.

- **Project Storage:** This is the shared space for all users in your project, which you are estimating and paying for.
- **Scratch Storage:** Available at no cost with a limit of 100TB per user (may be subject to change).
  - **Warning:** Scratch storage is intended for temporary staging purposes only. It is subject to a strict 30-day purge policy.
- **Home Directory:** Each user receives a complimentary 50GB.

Reality Checks & Limits

Before submitting, ensure your requested numbers do not exceed the maximum resources available per year.

- **ASPIRE 2A (CPU):** Total RIE 60% Allocation per year = 517 Million CPU Core-hours.
- **ASPIRE 2A (GPU):** Total RIE 60% Allocation per year = 1.85 Million A100 GPU Card-hours.
- **ASPIRE 2A+ (GPU):** Total RIE 60% Allocation per year = 1.68 Million H100 GPU Card-hours.
- **Upcoming System (CPU):** Total RIE 60% Allocation per year = 785 Million CPU Core-hours.
- **Upcoming System (GPU):** Total RIE 60% Allocation per year = 8 Million H200 GPU Card-hours.

If your project requires only modest or small resources, consider using departmental or institutional resources instead.

National Supercomputing Centre

Thank You

Appendix 1: Overview of NSCC

Overview of NSCC's Supercomputers

>3.5x More Cores

5x More Compact

2x More GPUs

7x More Compute Power

***Comparisons to ASPIRE 1 supercomputer**

105,984 Cores	352 GPUs	476TB	25 PBytes	10 PBytes
CPU (AMD EPYC™ 7713) **800 Nodes**	Accelerated Nodes GPU (NVIDIA A100) **82 Nodes**	Total System Memory	Storage (Spinning + Nearline)	Scratch Disk

HPC Top 500 Ranking	#1	#2	#3	#4	**#90**	**#301**	**#429**
System	El Capitan	Frontier	Aurora	Eagle	**ASPIRE2A+**	**ASPIRE 2A (GPU)**	**ASPIRE 2A (CPU)**
Manufactured by	HPE Cray	HPE Cray	HPE Cray	Microsoft	**Nvidia DGX**	**HPE Cray**	**HPE Cray**
Total Pflops	1,742.00	1,353.00	1012.10	561.70	**14.20**	**3.33**	**2.58**

***Ranking as of Nov 2024**

Knowing Your Hardware (ASPIRE2A)

Full list of HPC cluster node types and specification:

Server	CPU Model	Cores per socket	Socket per server	Total Physical cores per server	Available RAM (DDR4)	GPUs
Standard Compute Node (768 nodes)	Dual-CPU AMD EPYC 7713	64	2	128	512 GB	No GPU
GPU compute node (64 nodes)	Single-CPU AMD EPYC 7713	64	1	64	512 GB	**4x Nvidia A100 40GB**
GPU AI Node (12 nodes)	Single-CPU AMD EPYC 7713	64	1	64	512 GB	**4xNvidia A100 40GB** (11TB nvme)
GPU AI Node (6 nodes)	Dual-CPU AMD EPYC 7713	64	2	128	1 TB	**8xNvidia A100 40GB** (14TB nvme)
Large memory node (12 nodes)	Dual-CPU AMD EPYC 7713	64	2	128	2 TB	No GPU
Large memory node (4 nodes)	Dual-CPU AMD EPYC 7713	64	2	128	4 TB	No GPU

Storage

HOME+Project (GPFS) ≈ 25 PB

Scratch (Lustre) ≈ 10 PB

Interconnect (Slingshot 10)

1x100G Link (CPU node, Large Memory Node)

2x100G Link (GPU Node)

Free 16 CPU core-hours with every 1 GPU card-hour requested.

Knowing Your Hardware (ASPIRE2A+)

ASPIRE2A+ Components	Specification
Compute Nodes	NVIDIA DGX SuperPOD™ - 40 Nodes of DGX H100 (See below)
Interconnect	NVIDIA Quantum 2 based NDR InfiniBand
Storage	Scratch ~ 2.5 PB, Home ~ 27.5 PB

DGX H100	Specification
CPU	Dual Intel Xeon Platinum 8480C Processors Total Cores = 2 x 56 Cores = 112 Cores
System Memory	2 TB
GPU	8 x NVIDIA H100 GPUs
GPU Memory	640 GB (80 GB on each GPU card)
Storage	8 x 3.84 TB NVMe drives
Network	4 x OSFP ports for 8 x NVIDIA® ConnectX®-7 Si Cards 8 x 400 Gb/s InfiniBand
NVSwitch	4 x 4th generation NVLinks that provide 900 GB/s GPU-to-GPU bandwidth
Operating System	DGX OS (Ubuntu 22.04)
Performance	FP64 - 272 teraFLOPS, TF32 (Tensor Core) - 7.9 petaFLOPS, FP8 - 32 petaFLOPS

Free 14 CPU core-hours with every 1 GPU card-hour requested.

Appendix 2: Example detailed calculations for GROMACS and LLM training

Example 1: GROMACS - CPU

1. GROMACS runs efficiently on either CPU-only or CPU+GPUs. In this example, we focus on **CPU-only runs**.
2. **Project Overview:** MD simulations of a **1.4 million atoms** molecular system were conducted **for 100 ns.** A total of **100 different configurations** (varying temperature, forcefield, pH, etc.) were tested on the same molecular system. This is a 1-year project.
3. Get performance numbers, ***P*** of your previous GROMACS runs across **single or multiple CPU nodes** in **ns/day** if possible. These values can be found in GROMACS output files.
4. Shown in table in the next slide, the total hours required to complete a 100 ns simulation of a 1.4M-atom system is calculated using:

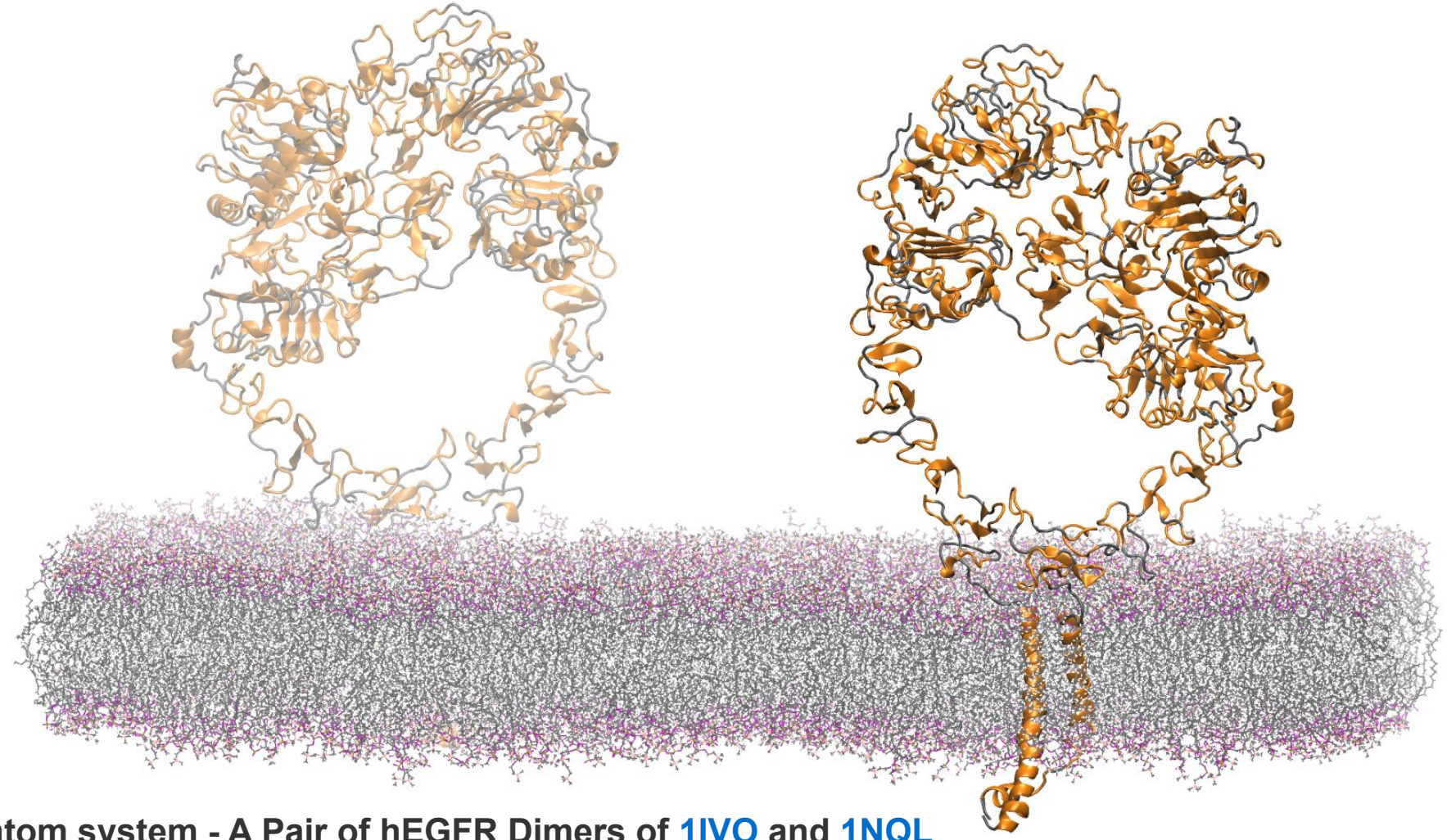
$$T = \frac{100 \times 24}{P} hours$$

5. To compute the required CPU core-hours, Σ_{CPU}:

$$\Sigma_{CPU} = T \times C$$

where ***C* = total number of cores**, ***T* = Time required for 100 ns simulation**, ***P* = Performance in ns/day, Σ_{CPU} = Total CPU Core-hour**

Example 1: GROMACS - CPU



1.4M atom system - A Pair of hEGFR Dimers of 1IVO and 1NQL

Total number of atoms = 1,403,182

Protein atoms = 43,498 Lipid atoms = 235,304 Water atoms = 1,123,392 Ions = 986

Credit: **HECBioSim**

Example 1: GROMACS - CPU

No. of Nodes (N)	No. of Cores (C)	Performance : ns/day (P)	Hours required for 100 ns simulation (T)	CPU core-hours required (Σ_{CPU})
1	128	5.716	420 (17.5 days)	53,744
2	256	11.343	212 (8.8 days)	54,166
4	512	20.308	118 (4.9 days)	60,508
8	1024	37.593	64 (2.7 days)	65,374
16	2048	63.246	38 (1.6 days)	77,716

The table above displays the variation in T for different numbers of C running the same simulation. Note that the P values shown here are based on NSCC's benchmarks. Users are encouraged to reference their own benchmarks when possible, as performance may vary across different datasets.

Example 1: GROMACS - CPU

6. We will use 77,716 CPU core-hours —rounded up to 80,000 CPU core-hours —for further calculations. Although requiring higher CPU core-hours. this choice is justified by the favorable scaling efficiency (69%), which enables **significantly reduced computation times**. In your project, shorter runtimes may be especially critical during the publishing stage or when responding to a journal reviewer's deadline.
7. The scaling efficiency S of the workload can be determined as:

$$S = \frac{P_{16}}{P_1 \times N_{16}}$$

where
- P_{16} is the performance of 16 CPU nodes,
- P_1 is the performance of 1 CPU node,
- N_{16} is the number of CPU nodes used in P_{16}.

This formula is adaptable—feel free to substitute 16 with any number of CPU nodes and performance values relevant to your setup.

8. **CPU Resources:** The total CPU core-hours required for this project would be 80,000 CPU core-hours × 100 runs = 8 Million CPU core-hours.

Example 1: GROMACS - CPU

9. **Storage Requirements:**
   - Storage estimation depends on the trajectory output frequency.
   - A single 100-ns run, saving trajectories every 100 ps (i.e., 1000 frames), may generate 1.5–3 GB of data in compressed XTC format .
   - For 100 runs, using the upper limit, this amounts to ~300 GB.
   - It is **best** to refer to past workload output sizes and extrapolate storage requirement based on trajectory frequency, types of information saved (coordinates, velocities, forces, etc.) simulation time.
10. **Buffer:** Include a 10% reserve for debugging, restarts, and unforeseen events.

Resources Required for this Project

1-Year Projection	CPU Core-hours	Project Storage (GB)
Estimated	8,000,000	300
Final Amount (With 10% Buffer)	8,800,000	330

Example 1: GROMACS - CPU

11. To calculate the monetary value of the resources requested, multiply the quantity by the advertised rate for each respective category. The following example uses the rates applicable to RIE-funded projects.

1-Year Projection	CPU Rate = S$ 0.006 per CPU core-hour	Storage GB-month = S$ 0.021
Final Compute Amount (With 10% Buffer)	8,800,000 CPU Core-hours	330 GB
Budget Required (RIE-Funded Pricing)	8,800,000 x S$ 0.006 = S$ 52,800	330 * 12 Months * S$ 0.021 = S$ 83.16

Total Amount = S$ 52,800 + S$ 83.16 =
S$ 52,883.16

Example 1: GROMACS - CPU

Important Disclaimer:

The resource estimates in this guide are derived from an online benchmark package that employs a specific force field and set of molecular constraints. These parameters may differ significantly from those in your actual workload. For the most accurate resource planning, we recommend basing your estimates on historical performance data from your own optimised runs.

In GROMACS, under typical simulation conditions, we observe near-linear scaling of computational time with respect to the number of atoms. In contrast, other HPC applications—such as DFT codes, CFD solvers, and climate models—can exhibit a range of scaling behaviors (e.g., linear, quadratic, cubic, or complex) depending on the underlying algorithms and computational tasks.

Example 2: GROMACS - GPU

1. GROMACS runs efficiently on either CPU-only or CPU+GPUs. In this example, we focus on **GPU runs**.
2. **Project Overview:** MD simulations of a **3 million atoms** molecular system were conducted **for 100 ns.** A total of **100 different configurations** (varying temperature, forcefield, pH, etc.) were tested on the same molecular system. This is a 1-year project.
3. Get performance numbers, ***P*** of your previous GROMACS runs across **single or multiple GPU cards** in **ns/day** if possible. These values can be found in GROMACS output files.
4. Shown in table in the next slide, the total hours required to complete a 100 ns simulation of a 3M-atom system is calculated using:

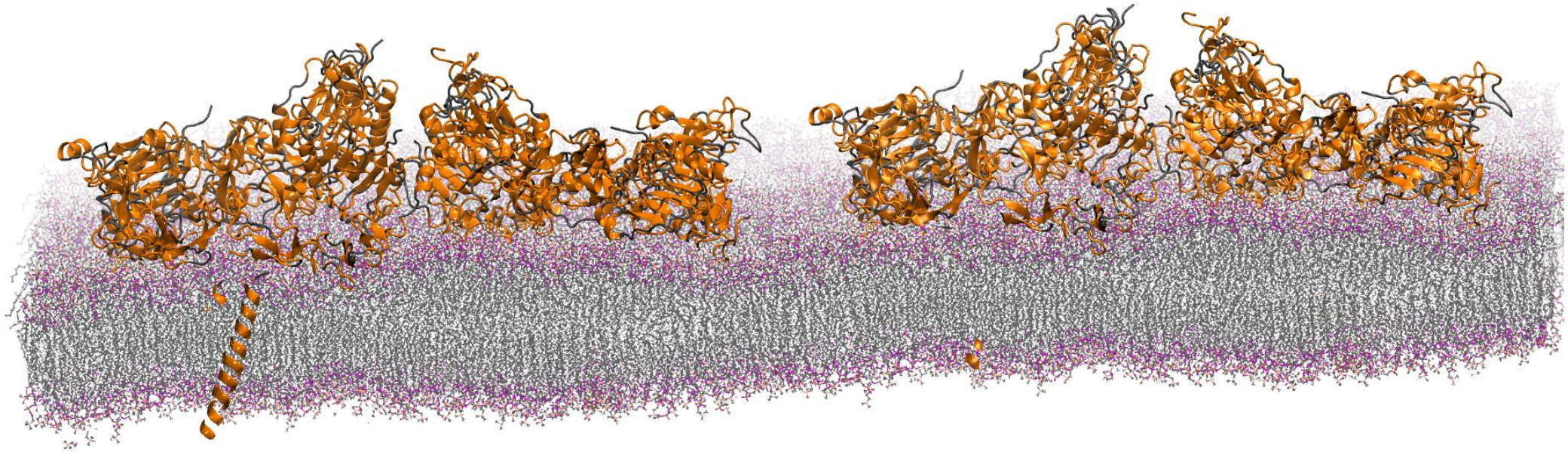
$$T = \frac{100 \times 24}{P} hours$$

5. To compute the required GPU card-hours (Σ_{GPU}):

$$\Sigma_{GPU} = T \times G$$

where ***G* = total number of GPU cards**, ***T* = Time required for 100 ns simulation**, ***P* = Performance in ns/day, Σ_{GPU} = Total GPU card-hour**

Example 2: GROMACS - GPU



3M atom system - A Pair of hEGFR tetramers of 1IVO and 1NQL

Total number of atoms = 2,997,924

Protein atoms = 86,996 Lipid atoms = 867,784 Water atoms = 2,041,230 Ions = 1,914

Credit: **HECBioSim**

Example 2: GROMACS - GPU

No. of GPUs (N)	Performance: ns/day (P)	Hours required for 100 ns simulation (T)	GPU card-hours required (Σ_{GPU})
1	6.453	372 (15.5 days)	372
2	10.671	225 (9.4 days)	450
4	17.866	134 (5.6 days)	536

The table above displays the variation in T for different numbers of N running the same simulation. Note that the P values shown here are based on NSCC's benchmarks. Users are encouraged to reference their own benchmarks when possible, as performance may vary across different datasets.

Example 2: GROMACS - GPU

6. We will use 536 GPU card-hours —rounded up to 600 GPU card-hours — for further calculations. Although requiring higher GPU core-hours. this choice is justified by the favorable scaling efficiency (69%), which enables **significantly reduced computation times**. In your project, shorter runtimes may be especially critical during the publishing stage or when responding to a journal reviewer's deadline.
7. The scaling efficiency S of the workload can be determined as:

$$S = \frac{P_4}{P_1 \times N_4}$$

where
- P_4 is the performance of 4 GPU cards,
- P_1 is the performance of 1 GPU card,
- N_4 is the number of GPU cards used in P_4.

This formula is adaptable—feel free to substitute 4 with any number of GPU cards and performance values relevant to your setup.

8. **GPU Resources:** The total GPU core-hours required for this project would be 600 * 100 runs **= 60,000 GPU card-hours.**

Example 2: GROMACS - GPU

9. **CPU core-hours:** Some CPU core-hours are required for tasks that run solely on the CPU, such as data transfer and compression. In this example, we allocate **100,000 CPU core-hours**. However, if additional CPU core-hours are needed for other tasks, please estimate them using the same approach as in **Example 1**.

9. **Storage Requirements:**
   - Storage estimation depends on the trajectory output frequency.
   - A single 100-ns run, saving trajectories every 100 ps (i.e., 1000 frames), may generate 3.0 – 6.0 GB of data in compressed XTC format .
   - For 100 runs, using the upper limit, this amounts to **~600 GB**.
   - It is **best** to refer to past workload output sizes and extrapolate storage requirement based on trajectory frequency, types of information saved (coordinates, velocities, forces, etc.) and simulation time.

10. **Buffer:** Include a 10% reserve for debugging, restarts, and unforeseen events.

Example 2: GROMACS - GPU

1-Year Projection	CPU Core-hours	GPU Card-hours (A100)	Project Storage (GB)
Estimated Compute	100,000	60,000	600
Final Compute Amount (With 10% Buffer)	110,000	66,000	660
Budget Required (RIE-Funded Pricing)	110,000 x S$ 0.006 = S$ 660	66,000 x S$ 0.79 = S$ 52,140	660 * 12 Months * S$ 0.021 = S$ 166.32

Total Amount = S$ 660.00 + S$ 52,140.00 + S$ 166.32 =

S$ 52,966.32

Example 2: GROMACS - GPU

- On our HPC system, the available GROMACS modules do not support multi-node GPU execution. Even if users compile and install a custom version of GROMACS that includes multi-node GPU support (requiring that cuFFTMp is properly installed) our estimates indicate that significant performance gains are only achievable for very large systems (i.e., those with tens or hundreds of millions of atoms).

- For smaller systems, scaling a single simulation across multiple GPU nodes often leads to diminishing, or even negative, performance returns due to communication overhead and load imbalance.

- Not all GROMACS simulations are fully GPU-capable. Depending on the GROMACS version and build configuration, certain features—such as specific implementations of virtual sites, external packages or custom force fields—may not be fully supported for GPU acceleration. Always verify that the particular simulation features you require are compatible with the GPU offload mode in your version of GROMACS

- Solution: Go to CPU nodes or wait for newer versions of GROMACS

Example 3: AI Project

1. **AI Project:** Training of a 10 B-parameter LLM on 100 B tokens (1 Epoch)
2. **References:**
   I. Training Compute-Optimal Large Language Models by *Hoffmann et. al.* (2022) – https://arxiv.org/abs/2203.15556
   II. Maximizing training throughput using PyTorch FSDP (MFU)

- *The resource estimates in this guide are derived from an FP16, full-parameter training example and are intended solely as an illustrative reference. They are **NOT** universally applicable.*
- *For example, DeepSeek employs a different architecture (MoE) and leverages alternative precision methods (i.e., FP8, BF16 and FP32 mixed precision), which result in different resource requirements.*
- *Please adjust these guidelines to reflect the specific training methodology and precision strategy of your model. (No. of Epoch, parameters, tokens, etc.)*

Example 3: AI Project

3. **Using Chinchilla Scaling: $C \approx C_0 * N * D$**, where
   - **C** = The Cost of training the model in FLOP
   - **C_0** = The statistical parameter = 6
   - **N** = The number of parameters in the model
   - **D** = The number of tokens in the training set
4. **Cost of training (FLOP), Hardware Performance (FLOP/s) and MFU**
   - **C** ≈ $6 * 10^{10}$ FLOP/token * 10^{11} tokens = 6×10^{21} FLOP (Floating point operations)
   - **P** (FP16 Performance in FLOPS) of Nvidia A100 ≈ roughly 3.12×10^{14} FLOPS (FLOP per Second)
   - **MFU** (Model FLOPS utilisation) ≈ 50%. This means the model is using roughly 50% of Nvidia A100 performance (**P**)

Example 3: AI Project

5. The compute time formula is $\mathbf{T} = \frac{C}{P*MFU}$
6. **Compute time, T** on one A100 GPU, to fully train a LLM with 10B parameters + 100B Tokens:

$$T_{1GPU} = \frac{C}{P*MFU} = \frac{6 \times 10^{21}\text{FLOP}}{(3.12\ 10^{14}\text{ FLOPS} \times 0.5)} \approx \textbf{11,000 Hours}$$

Compute Required = 1 GPU * 11,000 Hours = **11,000 GPU card-hours**

7. **CPU Resources:** Roughly **100,000 CPU core-hours** for pre- and post-processing (Tokenisation, data cleaning/compressing, etc.)
8. **Storage:** About **2 TB** (assuming 100 billion tokens × 4(or 2) bytes/token ≈ 400 GB, with additional overhead + Model Checkpoint).
9. **Buffer:** Include a **10%** reserve for debugging, restarts, and unforeseen events.

Example 3: AI Project

10. Training Duration when Parallelised (For Project Timeline Planning) :

- $T_{8\ GPUs}$ = 11,000 hours/8 = 1,375 hours ≈ 58 days.
- $T_{16\ GPUs}$ = 11,000 hours/16 = 687.5 hours ≈ 29 days.
- $T_{32\ GPUs}$ = 11,000 hours/32= 344 hours ≈ 15 days.
- $T_{64\ GPUs}$ = 11,000 hours/64= 172 hours ≈ 8 days.

*All days are rounded up to the nearest whole day

VRAM info is not required in project applications but is crucial for your model. Please ensure your model fits within available GPU memory and within your budget.

Resources Required for this Project

1-Year Projection	CPU (Core-hours)	GPU (A100 Card-hour)	Project Storage (GB)
Estimated	100,000	11,000	2,000
Final Amount (With 10% Buffer)	110,000	12,100	2,200

Example 3: AI Project

11. **PRICING**

1-Year Projection	CPU Core-hours	GPU Card-hours (A100)	Project Storage (GB)
Final Compute Amount (With 10% Buffer)	110,000	12,100	2,200
Budget Required (RIE-Funded Pricing)	110,000 x S$ 0.006 = S$ 660	12,100 x S$ 0.79 = S$ 9,559	2200 * 12 Months * S$ 0.021 = S$ 554.4

11. Total Amount = S$ 660.00 + S$ 9,559.00 + S$ 554.40 = **S$ 10,773.40**

11. Using the estimation method shown in the previous slides, training Llama 3.1 (405 billion parameter + 15 trillion tokens) would have costed 33.75M H100 GPU card-hours. *(Based on H100 post-MFU perf of 400 TFLOPS)*

11. The actual reported GPU card-hours used to train Llama 3.1 is 30.84M GPU card hours (H100 80 GB). This demonstrates the accuracy of the estimation method presented. (Using RIE rate, it will be S$ 42.53M vs S$ 38.86M)

Example 3: AI Project

15. Quiz: Why is it impossible to train Llama 3.1 on ASPIRE2A+ which has 320 H100 80GB GPUs?

30,840,000 GPU card-hours / 320 GPUs ≈ 11 Years

Meta trained them on <u>16K GPUs simultaneously</u>

16. You can also work backward from your compute budget by estimating the amount of FLOPs you can purchase. Using the scaling law C ≈ 6*N*D and the heuristic that the optimal training regime uses roughly ±20 tokens per parameter (i.e. D ≈ 20N), you can determine the maximum model size (N) and dataset size (D) that your budget supports.

Useful Links for Resource Estimation

Nvidia Benchmark (AI)– GPU

1. https://developer.nvidia.com/deep-learning-performance-training-inference/training
2. https://developer.nvidia.com/deep-learning-performance-training-inference/ai-inference
3. https://developer.nvidia.com/deep-learning-performance-training-inference/conversational-ai
4. https://docs.nvidia.com/nemo-framework/user-guide/24.09/performance/performance_summary.html

AI Training or Fine-tuning Resource Guideline – GPU

1. DeepSeek-V3 Technical Report
2. https://docs.api.nvidia.com/nim/reference/meta-llama-3_1-405b
3. Understanding the Performance and Estimating the Cost of LLM Fine-Tuning
4. https://en.wikipedia.org/wiki/Neural_scaling_law

Nvidia Benchmark (HPC Applications) – GPU

1. https://developer.nvidia.com/hpc-application-performance

GROMACS, AMBER, NAMD, LAMMPS (HPC Application) – CPU and GPU

1. https://www.hecbiosim.ac.uk/access-hpc/our-benchmark-results/archer2-benchmarks
2. https://www.hecbiosim.ac.uk/access-hpc/our-benchmark-results/bede-benchmarks