Using HPC systems for Python-based AI/ML tasks

UM Spring HPC/Cloud Workshop

Grigory Shamov, May 22, 2025

TRADITIONAL TERRITORIES ACKNOWLEDGEMENT

The University of Manitoba campuses are located on original lands of Anishinaabeg, Cree, Oji-Cree, Dakota, and Dene peoples, and on the homeland of the Métis Nation.

We respect the Treaties that were made on these territories, we acknowledge the harms and mistakes of the past, and we dedicate ourselves to move forward in partnership with Indigenous communities in a spirit of reconciliation and collaboration.

Why the talk is about Python AI/ML on HPC?

- **Everyone (almost) is using Python for AI / ML**
- **HPC has high performance hardware and software**
  - NVIDIA (and AMD) GPUs
  - High bandwidth, Low-latency Interconnect (NVidia Infiniband)
  - Scalable, parallel File Systems (VAST, CEPH, DDN Infinia) or local SSD NVMe
- **HPC community has a datacentre infrastructure**
  - Power; high efficiency cooling
- **Many AI shops do use same HPC technology**
  - **SLURM, Linux, etc**.
  - Some do use Kubernetes and containerized workflows

This talk is about using our HPC infra

- HPC presents a few issues for running AI workflows:
    - How to maintain the (largely Python-based) software on HPC?
        - Need to be able to get software and AI Models running
    - Good to have some ability to interactive workflows for debugging
    - How to adapt it to the SLURM based, batch workflow
        - Need to match HPC resources with AI workflow efficiently
- SaaS competition! (Google Colab, for example).

Artificial Intelligence and Machine Learning revolution

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/35179.pdf "The Unreasonable Efficiency of Data"

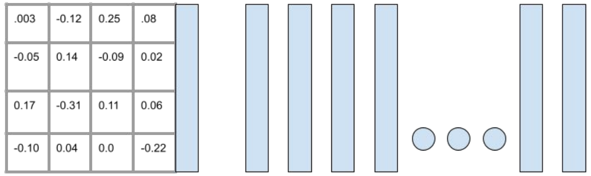
http://www.incompleteideas.net/IncIdeas/BitterLesson.html "The Bitter Lesson"

https://dl.acm.org/doi/10.5555/3295222.3295349 "Attention is all you need"

https://blogs.nvidia.com/blog/what-are-foundation-models/ NVidia on foundational models.

What are those AI and “models”

- Layers of numerical weights, and metadata about them
- Specific to the model’s software and hardware (precision)
  - (“transformers” would have attention layer, etc.)



.003	-0.12	0.25	.08
-0.05	0.14	-0.09	0.02
0.17	-0.31	0.11	0.06
-0.10	0.04	0.0	-0.22

- **Training/tuning** optimizes the weights of all or some layers
- **Inference** converts a prompt into a numerical representation , passes through model layers and generates outputs

https://poloclub.github.io/transformer-explainer/

Working with AI software and “models”

- Python packages : matrix math and optimization (*torch*, *arrow*, *JAX*, …), ML models (*transformers*, *diffusers*, …) , training (*accelerate*, *LoRa*, etc.)
  - Need to manage software dependencies (modules, pip, virtualenv)
  - Must use optimized libraries from the host (CUDA, MPI, BLAS/LAPACK, Arrow) → modules
  - Common tools like *conda* are often unwelcome in HPC
- The AI Models themselves (layers, weights) need to be obtained

- A popular community hub Huggingface : https://huggingface.co/
- https://docs.alliancecan.ca/wiki/Huggingface

Typical tasks for AI / ML on HPC

- Training new models : optimize the weights
- Tuning existing models on additional datasets (LoRA, etc., but more economical: selected layers)
- Inference: getting predictions/generalizations out of existing models
  - Tokenize the “prompt”,
  - Feed it through the layers
  - Convert the representation back to the “output”

- For this Session we will try three examples: Interactive inference for text, text to graphics and tuning/training a graphic model in a batch job.
- Using Grex (mainly) or Magic Castle: need GPUs!

Managing Python dependencies with pip

- Dependencies: Pip handles (mostly) Python dependencies, relying on OS for non-Python ones
  a. Unlike conda, uv etc, that would package all binary dependencies as well.
  b. HPC folks like to provide their own optimized binary software : load “modules”!
- Repositories: pip fetches packages from https://pypi.org/ or a local repository.
  a. On CCEnv, each and every package must be repackaged to their “wheelhouse”.
  b. On local software stack SBEnv, manylinux wheels can be used from pypi.org
- Installation destination: in particular when invoked in a Jupyter notebook cell.
  a. user’s home directory $HOME/.local/{bin,lib,share} . (Grex)
  b. Throw-away virtual environment under $SLURM_TMPDIR (Alliance HPC , MagicCastle)
  c. Explicit virtualenv (create, activate and install) : best method. Needs a Jupyter kernel added.

1. Project one: text generation using Qwen3

- We will generate text based on a prompt, with reasoning using a small LLM
- https://huggingface.co/Qwen/Qwen3-0.6B
- The code has three general stages
  1. But first, use pip to install packages in Jupyter.
     1.1. Don’t forget to load Lmod modules (Lmod tab) !
     1.2. Monitoring tab for Node and GPU resources.
  2. Import Python packages and see if they load
  3. Set up encoders for the prompt
  4. Run the inference/generation step
  5. Decode the output, print result
- Can be made into a chatbot with tools like Gradio.

2. Project two: inference using SD-1.5

- We will generate images based on a text prompt
- https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5
  - (used to be runwayml/stable-diffusion-v1-5)
  - Will use *torch*, *transformers* and *diffusers*
  - Needs a GPU! On Grex or MagicCastle
  - Will use HuggingFace convenient “*pipe*” interface that lets us set a pretrained model and abstract the encoding, decoding steps.
- Let us use same pip environment in Jupyter as for the project 1.

3. Project 3: using LoRa to train SD-1.5

- The SD-1.5 model is unaware of certain things. I want to add some knowledge to it: an ElderScrolls game character, Scamp
  - https://en.uesp.net/wiki/Lore:Scamp
  - Will use a handful of images from the fandom wiki (dataset 2)
  - Will use a handful of auto generated images from ChatGPT 4o. (dataset 1)
  -
- Low Rank Adaptation of Large Language Models (LoRa) technique
  - Only adding a few layers, freezing the rest of the model
  - https://huggingface.co/docs/diffusers/en/training/lora

```
unet_lora_config = LoraConfig(
    r=args.rank,
    lora_alpha=args.rank,
    init_lora_weights="gaussian",
    target_modules=["to_k", "to_q", "to_v",
"to_out.0"],
)

unet.add_adapter(unet_lora_config)
lora_layers = filter(lambda p:
p.requires_grad, unet.parameters())
```

3. Project 3: using LoRa to train SD-1.5

- Using Huggingface “*accelerate*” training script from “*diffusers*” repo
    - Get an interactive job on a GPU node
    - Git clone the repo TBD
    - Create a virtualenv (SBEnv on Grex, CCEnv on Jupyter)
    - Install pip packages, including current versions of *diffusers* from git
    - Download / copy the training data
        - Dataset 1 and dataset 2 with “Scamps”.
    - Run a salloc training job, obtaining new LoRa weights.
    - Validate the model, repeat etc.

3. Project 3: using LoRa to train SD-1.5

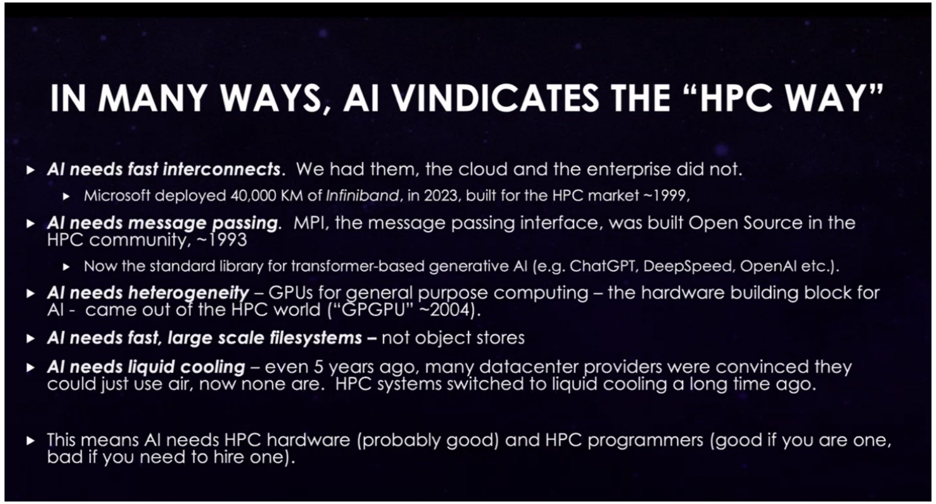
- Using Jupyter, repeat the exercise 2 but use generation with new layers added.
  - Will use HuggingFace convenient “*pipe*” interface that lets us set a pretrained model and abstract the encoding, decoding steps.
  - Let us use same pip environment in Jupyter as for the project 2.
  - The only change is to add our new extra layers to the model.
  - Try to generate an image about a “Scamp”.

Strategies for larger AI Model training

- Large datasets, take a lot of time on a single GPU
- Large models would not fit into memory of a single GPU
- Distributed parallel training frameworks: .
  - Huggingface *accelerate* ,
  - Microsoft *DeepSpeed* ,
  - etc,
- Data parallelism (split the data across GPUs)
- Pipeline parallelism (distribute model layers)
- Tensor parallelism (domain decomposition)

May rely on collective MPI operations (AllReduce, AllGather etc,)

DRI (Digital Research Infrastructure) for AI



IN MANY WAYS, AI VINDICATES THE "HPC WAY"

- ***AI needs fast interconnects***. We had them, the cloud and the enterprise did not.
  - Microsoft deployed 40,000 KM of *Infiniband*, in 2023, built for the HPC market ~1999,
- ***AI needs message passing***. MPI, the message passing interface, was built Open Source in the HPC community, ~1993
  - Now the standard library for transformer-based generative AI (e.g. ChatGPT, DeepSpeed, OpenAI etc.).
- ***AI needs heterogeneity*** – GPUs for general purpose computing – the hardware building block for AI - came out of the HPC world ("GPGPU" ~2004).
- ***AI needs fast, large scale filesystems*** – not object stores
- ***AI needs liquid cooling*** – even 5 years ago, many datacenter providers were convinced they could just use air, now none are. HPC systems switched to liquid cooling a long time ago.
- This means AI needs HPC hardware (probably good) and HPC programmers (good if you are one, bad if you need to hire one).

A slide by Dan Stazione, Director of TACC

***tamIA**, a real AI supercomputer of LavalU*

Which system for which workload?

Digital Research Alliance of Canada

System, kind (2025)	# GPU nodes	GPUs per node layout	Interconnect	Storage, PB
TamIA , HPC (Laval)	42 (H100)	**4 x** NVIDIA HGX H100 SXM	**4 x** HDR200 Infiniband, **non-blocking**	?
Vulcan, HPC (UofA)	205 (L40s)	4 x NVIDIA L40s	1x100Gbps Ethernet	5PB
Killarney, HPC (UofT)	168 (L40s) 10(H100)	4 x NVIDIA L40s, 8 x NVIDIA H100 SXM	1x HDR100, 2x HDR200	1.5 PB
HPC systems ?				
Fir, HPC	160	4 x NVidia H100 SXM	1x HDR200 Infiniband, blocking	51PB
Nibi, HPC	36	8 x Nvidia H100 SXM	1x Nokia 200/400G Ethernet	25PB
Trillium, HPC	60	4 x NVidia H100 SXM	1x NDR200/ NDR400 Infiniband	29PB

University of Manitoba