INTRODUCTION TO AI TESTBEDS AT ALCF AND HANDS-ON

SIDDHISANKET (SID) RASKAR
Postdoctoral Researcher
Argonne Leadership Computing Facility
Argonne National Laboratory

August 11th, 2023
St. Charles, IL
ALCF AI Testbed

https://www.alcf.anl.gov/alcf-ai-testbed

- Cerebras: 2 CS-2 nodes, each with 850,000 Cores, compute-intensive models
- SambaNova: DataScale SN30 8 nodes (8 SN30 RDUs per node) - 1TB mem per device, models with large memory footprint
- Graphcore: BowPod64 4 nodes (16 IPUs per node) - MIMD, irregular workloads such as graph neural networks
- GroqRack: 8 nodes, 8 GroqNodes per node - inference at batch 1
- Habana Gaudi1: 2 nodes, 8 cards per node - On-chip integration of RDMA over Converged Ethernet (RoCE2), scale-out efficiency
ALCF AI Testbed

https://www.alcf.anl.gov/alcf-ai-testbed

Track 1 – Hardware Architectures

Advancing Scientific Machine Learning with AI Accelerators in ALCF AI Testbed

Murali Emani

Learn about Architectures

Learn about Dataflow

Track 1 – Hardware Architectures

Introduction on DataFlow Architectures and Trends

Jose Monsalve Diaz, Sid Raskar

Cerebras CS-2

SambaNova DataScale SN30

Graphcore Bow Pod64
Director’s Discretionary (DD) awards support various project objectives from scaling code to preparing for future computing competition to production scientific computing in support of strategic partnerships.

Getting Started on ALCF AI Testbed:

Apply for a Director’s Discretionary (DD) Allocation Award

Cerebras CS-2, SambaNova Datascale SN30 and Graphcore Bow Pod64 are available for allocations

Allocation Request Form

AI Testbed User Guide
CEREBRAS WAFER-SCALE ENGINE (WSE-2)

Still the Largest Chip Ever Made

850,000 cores optimized for sparse linear algebra
46,225 mm² silicon
2.6 trillion transistors
40 gigabytes of on-chip memory
20 PByte/s memory bandwidth
220 Pbit/s fabric bandwidth
7nm process technology

Cluster-scale performance in a single chip
CEREBRAS SYSTEMS AT ALCF

- 2-node Wafer-Scale Cluster
  - Supporting up to 30BN parameter models
  - Computer Vision and NLP optimized
- 2x CS-2s, with:
  - 850k cores each
  - 40GB on chip memory each
- Can distribute jobs across one or both CS-2s, with data parallel scaling when using both machines
CONNECTION TO A CS-2 NODE

https://docs.alcf.anl.gov/ai-testbed/cerebras/getting-started/

Log in to Login Node

$ ssh ALCFUserID@cerebras.ai.alcf.anl.gov
$ Password: <MobilePass+ Code>
ENVIRONMENT SETUP

Cerebras virtual environments

$ /software/cerebras/python3.8/bin/python3.8 -m venv venv_pt
$ source ~/venv_pt/bin/activate
$ pip3 install --disable-pip-version-check /opt/cerebras/wheels/cerebras_pytorch-1.9.1+1cf4d0632b-cp38-cp38-linux_x86_64.whl --find-links=/opt/cerebras/wheels/
$ pip install numpy==1.23.4
$ pip install datasets transformers

On subsequent logins

$ source venv_pt/bin/activate
WORKFLOW

• **Compile**
  • Compiles are done automatically when no usable cached compile is found for the model.
  • Maps the resources required to run an application to a CS-2 wafer.
  • Significant compile times for large models.

• **Run**
  • Execution of a compiled model using
    • One or more CS-2s.
    • Support nodes in the CS-2 cluster.
Clone Cerebras Modelzoo

```bash
$ mkdir ~/R_1.9.1
$ cd ~/R_1.9.1
$ git clone https://github.com/Cerebras/modelzoo.git
$ cd modelzoo
$ git tag
$ git checkout Release_1.9.1
```

https://github.com/Cerebras/modelzoo.git
# TYPICAL ANATOMY OF A MODEL IN MODEL ZOO

<table>
<thead>
<tr>
<th>run.py</th>
<th>Main script to execute train, eval, prediction in CS-2 or GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>configs/</td>
<td>Folder with different parametrizations of the model in .yaml files</td>
</tr>
<tr>
<td>model.py</td>
<td>Creation of the NN model function</td>
</tr>
<tr>
<td>utils.py</td>
<td>Helper functions to set up run.py</td>
</tr>
<tr>
<td>data.py</td>
<td>Helper functions to prepare data</td>
</tr>
</tbody>
</table>
EXAMPLE PROGRAM – MNIST / PIPELINED

Goto Example Directory

$ cd ~/R_1.9.1/modelzoo/modelzoo/fc_mnist/pytorch/

Activate the environment.

$ source venv_pt/bin/activate

Compile and Run

$ export MODEL_DIR=model_dir
$ if [ -d "${MODEL_DIR}" ]; then rm -Rf $MODEL_DIR; fi

$ cp /software/cerebras/dataset/fc_mnist/pytorch/configs/params.yaml.modified configs/

$ python run.py CSX pipeline --job_labels name=pt_fc_mnist --params configs/params.yaml.modified --mode train --model_dir $MODEL_DIR --mount_dirs /home/ /software --python_paths /home/$(whoami)/R_1.9.1/modelzoo --compile_dir /$(whoami) |& tee mytest.log
### IMPORTANT DIRECTORY PATHS AND LINKS

**Cerebras Modelzoo Repository**

https://github.com/Cerebras/modelzoo.git

**Important datasets Path**

/software/cerebras/dataset

---

**AI Testbeds User Guide**

**Cerebras Documentation**
SAMBANOVA CARDINAL SN30 RDU

- 7nm TSMC, 86B transistors
- 102 km of wire
- 640 MB on-chip, 1,024 GB external
- 688 TFLOPS (bf16)

Cardinal SN30™
Reconfigurable Dataflow Unit™

RDU-Connect™
CONNECTING TO SAMBANOVA SN-30

Log in to Login Node

$ ssh ALCFUserID@sambanova.alcf.anl.gov
Password: <MobilePass+ Code>

From login node to a SN30 node

$ ssh sn30-r1-h1
ENVIRONMENT SETUP

SambaFlow software stack and the associated environmental variables are setup at login

Create Virtual Environment and Install Packages

$ python -m venv --system-site-packages my_env
$ source my_env/bin/activate

$ python3 -m pip install <package>

Pre-Built Environments

/opt/sambaflow/apps/
WORKFLOW

Compile

- Model compilation and '.pef' generation.
- Maps the compute and memory resources required to run an application on RDUs
- Re-compile only when model parameters change.
- Significant compile times for large models.

```bash
srun python lenet.py compile -b=1 --pef-name="lenet" --output-folder="pef"
```

Run

- Model trained on RDU using the '.pef' generated as part of compile process and the training dataset.

```bash
srun python lenet.py run --pef="pef/lenet/lenet.pef"
```
EXAMPLE PROGRAM: MNIST

Make a copy of the apps directory into the home directory

```
$ cp -r /opt/sambaflow/apps/ ~
```

Activate Virtual Environment

```
$ source ~/apps/starters/ffn_mnist/vene/bin/activate
```

Compile and Run

```
$ srun python ffn_mnist.py compile -b=1 --pef-name="ffn_mnist" --mac-v2
$ srun python ffn_mnist.py run -b 1 -p out/ffn_mnist/ffn_mnist.pef
```
<table>
<thead>
<tr>
<th>IMPORTANT DIRECTORY PATHS AND LINKS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sambanova Applications Path</strong></td>
</tr>
<tr>
<td>/opt/sambaflow/apps/</td>
</tr>
<tr>
<td><strong>Sambanova Model scripts</strong></td>
</tr>
<tr>
<td>/data/ANL/scripts/</td>
</tr>
<tr>
<td><strong>Important Datasets</strong></td>
</tr>
<tr>
<td>/software/sambanova/dataset/</td>
</tr>
</tbody>
</table>

**AI Testbeds User Guide**

**Sambanova Documentation**
CONNECTION AND LOGIN

Log in to Login Node

$ ssh ALCFUserID@gc-login-01.ai.alcf.anl.gov
$ Password: <MobilePass+ Code>

Log in to a Graphcore Node

ssh gc-poplar-02.ai.alcf.anl.gov
ENVIRONMENT SETUP

The Poplar SDK on Graphcore

/software/graphcore/poplar_sdk/

The default poplar version (3.1.0) is enabled automatically upon logging into a graphcore node.

PopTorch Environment Setup

$ mkdir -p ~/venvs/graphcore
$ virtualenv ~/venvs/graphcore/poptorch31_env
$ source ~/venvs/graphcore/poptorch31_env/bin/activate
$ export POPLAR_SDK_ROOT=/software/graphcore/poplar_sdk/3.1.0
$ export POPLAR_SDK_ROOT=$POPLAR_SDK_ROOT
$ pip install $POPLAR_SDK_ROOT/poptorch-3.1.0+98660_0a383de63f_ubuntu_20_04-cp38-cp38-linux_x86_64.whl
## EXAMPLE PROGRAMS

### Clone Graphcore Examples Repository

```bash
$ git clone https://github.com/graphcore/examples.git
$ cd examples
```

### Activate PopTorch Environment for MNIST and install dependencies

```bash
$ cd examples/tutorials/simple_applications/pytorch/mnist
$ python -m pip install torchvision==0.14.0
```

### Run MNIST Example

```bash
$ /opt/slurm/bin/srun --ipus=1 python mnist_poptorch.py
```
IMPORTANT DIRECTORY PATHS AND LINKS

**Graphcore Examples Repository**
/opt/sambaflow/apps/

**Graphcore SDK Path**
/software/graphcore/poplar_sdk

**AI Testbeds User Guide**

**Graphcore Documentation**
ACKNOWLEDGMENTS

Bill Arnold
Varuni Sastry
Zhen Xie
Murali Emani