HTCondor in Foxconn Hyperscale AI HPC Cloud

Cheng-Han Du, Ph.D.
Foxconn Corporation
• Foxconn AI HPC Cloud Solution
  • Introduction
  • AI/HPC Cloud Stack – Hardware & Software
  • Applications

• HTCondor in Foxconn Cloud
  • Research Case – Bayesian Optimization of Deep Learning Hyperparameters
- Collaborate with Microsoft: “TECHNOLOGY OF THE YEAR” award of InfoWorld in 2018
- Solution in multiple scales
  - HGX-1, HGX-2, and workstations: standalone usage.
  - M100 Rack: All-in-one HPC box for on-premise solution
  - M500 Cloud: Ultimate scalable HPC system for public cloud
    - Serving computations of wide variety of scales
    - Massively scalable performance (HPC)
    - Huge number of smaller-scale and independent tasks for many customers (HTC)
Objectives

Democratizing HPC
- Reduce usage difficulty of HPC resource
- Users focus more on application, less on IT issues

Multidisciplinary Collaboration
- Close collaboration with customers and domain experts
- Cloud deployment consulting for customer
- Codesign of multiple applications: numerical simulation and AI with HPC

Streamlined for Deployment
- Utilize big data IT, cloud platform
- ML/DL/HPC as a service

Foxconn AI HPC Solution
### Applications

1. High-Resolution (8K) Image/Video Object Detection
2. Industrial Application / Surface Defect Detection
3. Medical Application / Deep learning / Industrial AI / NGS
Machine Learning / Deep Learning Service

1. Multi-node multi-GPU cluster environment
2. Training and inference workflow
3. Deep learning inference application platform
4. Performance autotuning (FAUST, SOFA, POTATO)
Foxconn AI HPC Stack

HPC Service
1. Optimized job scheduler and virtualization environment according to specified performance-energy criteria
2. Job dispatch in hybrid HPC environment, including on-premise HPC clusters and public cloud data centers
Hybrid Cluster Management (In Progress)

- Coexist HTCondor, Slurm, and K8S cluster under an OpenStack tenant
- Allow Slurm, K8S, and HTCondor batch jobs to share the same cluster hardware resources
 HGX-1 Architecture

**Modular GPU computing solution**

- Reconfigurable *scale-up* system
- High-density GPU clusters for space saving
- CPU node
  - 32 cores in total

- Single-node with 8 or 16 GPUs

---

**HGX-1 Stingray-HPC64, Local M.2 SSD**

- **CPU:** Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz
- **Memory:** DDR4 2666MHz 32G x 24
- **GPU:** Tesla V100-SXM2-32GB
- **Driver:** 410.72
- **OS:** NGC nvcr.io/nvidia/tensorflow:18.09-py3
- **Cuda:** cuda 10
- **Cudnn:** 7.3.0.29-1
- **Tensorflow:** 1.10
- **Benchmark:** 1.9

**8 GPU**

<table>
<thead>
<tr>
<th>ResNet50</th>
<th>BS=64</th>
<th>BS=128</th>
<th>BS=256</th>
<th>BS=512</th>
</tr>
</thead>
<tbody>
<tr>
<td>fp32</td>
<td>2395</td>
<td>2760</td>
<td>2987</td>
<td>X</td>
</tr>
<tr>
<td>fp16</td>
<td>3959</td>
<td>5129</td>
<td>6029</td>
<td>6464</td>
</tr>
</tbody>
</table>

**16 GPU**

<table>
<thead>
<tr>
<th>ResNet50</th>
<th>BS=64</th>
<th>BS=128</th>
<th>BS=256</th>
<th>BS=512</th>
</tr>
</thead>
<tbody>
<tr>
<td>fp32</td>
<td>4448</td>
<td>5306</td>
<td>5847</td>
<td>X</td>
</tr>
<tr>
<td>fp16</td>
<td>7140</td>
<td>9459</td>
<td>11244</td>
<td>12362</td>
</tr>
</tbody>
</table>
Parabricks NGS

- NGS pipeline speedup: \(30 \sim 50X\)
- Sequencer-generated FASTQ files to variant call files (VCFs) with GPUs

---

**Performance**

<table>
<thead>
<tr>
<th>Baseline (32 vCPU)</th>
<th>4 GPU</th>
<th>8 GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1870</td>
<td>60</td>
<td>35</td>
</tr>
</tbody>
</table>


![Graph showing throughput in genomes per day for different numbers of GPUs.](image)
Medical Applications

Mammogram abnormality detection

- Single image upload identification
  - Detection of abnormal position and score

- Multiple database image recognition
  - Batch detection and anomaly location

- Batch process with 8 GPUs
  - 600,000+ images processed per day

Real-time instant identification

Reference:
2. Digital Mammography DREAM Challenge
# HGX-2 Architecture

<table>
<thead>
<tr>
<th>HGX-1</th>
<th>HGX-2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GPU # in single node</strong></td>
<td>8 (single box) or 16 (cascaded boxes)</td>
</tr>
<tr>
<td><strong>Inter-GPU communication</strong></td>
<td>NVLink ‘Cube Mesh’ + PCIe</td>
</tr>
<tr>
<td><strong>Inter-GPU performance</strong></td>
<td>1 or 2 hops in 8 GPUs of an expansion box, plus 2 hops across PCIe of different boxes</td>
</tr>
<tr>
<td><strong>Ideal applications</strong></td>
<td>Applications require moderate GPU data exchange, e.g. most DL training models, ResNet, AlexNet, etc.</td>
</tr>
<tr>
<td><strong>TCO per GPU</strong></td>
<td>Low</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>Now</td>
</tr>
</tbody>
</table>

GPU Infrastructure Consideration

**On-premise GPU clusters**
- Improved data security and application latency
- Limited computing resource
- High maintenance effort

**Off-premise GPU data center**
- Virtually unlimited computing resource for bursts of critical demands
- Low maintenance cost

**HTCondor**
- High throughput for massive number of tasks
- Configurable hardware demand for each application
  - **Docker universe**
    - Swift deployment of complex application container
- On-demand GPU computing
  - HTCondor Annex
  - VM instance management in cloud
- **Federation of clusters**
- **DAGMan**
  - Easy orchestration of tasks

Public Cloud

Cloud Burst Request w/ Container Management Platform

On-premise HPC Rack
• Foxconn AI HPC Cloud Solution
  • Company Introduction
  • AI/HPC Cloud Stack – Hardware & Software
  • Applications

• HTCondor in Foxconn Cloud
  • Research Case – Bayesian Optimization of Deep Learning Hyperparameters
Research Case

Task dispatch for hyperparameter optimization

Application
• Optimization of AI model or many scientific problems
• Optimization scheme
  • Grid search (parallel)
  • Random search (parallel)
  • Bayesian optimization (serial in general)

HTCondor Features
• GPU computing
• Docker universe
  • Default runtime set to “nvidia”
• Partitionable slots
• HTCondor-Python

Major Tools
• Scikit-optimize (Bayesian optimization)
• Image: nvcr.io/nvidia/tensorflow:18.12-py3 (Keras)

Reference: https://github.com/fmfn/BayesianOptimization/blob/master/examples/bayesian_optimization.gif
Task dispatch for hyperparameter optimization

Environments and Setting

- Single-node with 8 Tesla P100 in Foxconn/DCT data center in Kaohsiung
- Mini-HTCondor
  - Single worker with partitionable resources
- Submitter
  - Python environment with HTCondor API and scikit-optimize
  - Eight observation tasks submitted in each round of optimization
  - One GPU and 4 CPU slots for each task

Optimizer generates parameter sets → Submit observation tasks → Wait all task results → Final optimizer round?
Task dispatch for hyperparameter optimization

Test script
- Keras example: cifar10_cnn.py
  - 50 epochs
- 8 optimization rounds
- Generally underfitting
  - Baseline setting
    - Learning rate: $1.0 \times 10^{-4}$
    - Layer size: 512
    - Validation accuracy: 77.74%
  - Tuning range
    - Learning rate: $[1.0 \times 10^{-6}, 1.0]$
    - Layer size: $[128, 1024]$

Summary
- Best accuracy in the 5th round
- Suggested learning rate: $[0.00005, 0.00015]$
- Suggested layer size: $[512, 1024]$

Extension
- Scale-out for more observations in one round
- Good results could be obtained with less rounds
Task dispatch for hyperparameter optimization

Discussion & Further Investigation

- Network bandwidth
  - Training data upload for multiple tasks
  - ImageNet: over 100 GB of data
- Potential solutions
  - Lustre share storage
  - Proxy
  - Docker volume mount
    - Prepare data before optimization tasks start
    - Mount volume of prepared data
    - Pro: data sharing of multiple tasks in the same node
    - Caveat: data nodes v.s. worker nodes specified by central manager

- Data Security
  - Primary concern in commercial applications
  - Potential solution
    - Encrypted filesystem
Thank You!