Intelligent System for AI

清大資工 周志遠

2018/5/19 @ AII Workshop
• 周志遠 (Jerry Chou)
  – Email: jchou@cs.nthu.edu.tw
  – Large-scaled System Architecture (LSA) Lab

• 經歷
  – 清華大學資工系 副教授 2016~現今
  – 清華大學資工系 助理教授 2011~2016
  – 美國勞倫斯國家實驗室 工程師 2010~2011
  – 美國加州大學聖帝亞哥分校(UCSD) 博士學位 2009

• 研究領域
  – 雲端計算、分散式系統、高效能計算、巨量資料處理
AI

High throughput & cost effective

System

Service Interface
- Tenant A
- Tenant B
- Tenant C

Resource Orchestration
- Web Developer
- Big Data Analysis
- DNN Training

Hardware Virtualization
- VM
- baremetal
- container

Intelligent resource management & system administration

ResNet-50

256 GPUs in one hour

DGX-1

150,000 USD

[facebook2017]
**Systems for AI**

**Public cloud**

- Managed service
- Pay-as-you-used
- Availability, Reliability

**Cost:** 10K TWD for 256GPU-hour

Data privacy and transfer
Systems for AI

Public cloud
- Managed service
- Pay-as-you-used
- Availability, Reliability

Cost: 10K TWD for 256GPU-hour
Data privacy and transfer

Private cloud
- Control & efficiency
- Security & privacy
- Customization

Complex & virtualized HW infra.
Diverse SW deployment
Resource management
Systems for AI

- Public cloud
  - Managed service
  - Pay-as-you-used
  - Availability, Reliability
  - Cost: 10K TWD for 256GPU-hour
  - Data privacy and transfer

- Private cloud
  - Control & efficiency
  - Security & privacy
  - Customization
  - Complex & virtualized HW infra.
  - Diverse SW deployment
  - Resource management

【財訊快報／王宜弘報導】搶攻AI商機，台廠大團結！華碩(2357)、燦達(2382)以及台灣大(3045)結盟組成「台灣人工智慧A Team」，成軍後首戰告捷！週一(30日)三方共同宣布取得國家實驗研究院國家高速網路與計算中心「雲端服務及大數據運算設施暨整合式階層儲存系統建置案」，將協助建置新一代的AI計算主機，並建立產官學研共用具延展性的AI雲端大資料計算平台，建置總金額近11億新台幣，預計今年第四季建置完成。
Key Challenges of AI Systems

• System Infrastructure:
  – VM + CPU
  ➔ Container + GPU

• Training job execution:
  – Static Single instance execution
  ➔ Elastic distributed execution
Container-based GPU Cloud

• Why Container?
  – Lightweight, low performance overhead
  – High deployment density
  – Execution environment isolation

Benchmark TensorFlow on varied resource orchestration (baremetal, container, VM) and execution environment (single, distributed, multi-tenant)
Container-based GPU Cloud

- Why Container?
  - Lightweight, low performance overhead
  - High deployment density
  - Execution environment isolation

Container can deliver close to the bare-metal performance in dedicated resource environment
Container-based GPU Cloud

• Why Container?
  – Lightweight, low performance overhead
  – High deployment density
  – Execution environment isolation

• Container lacks QoS control for PCIE and GPU
  • GPU may not be fully utilized by a single job
Container-based GPU Cloud

• Why Kubernetes (container orchestrator)?
  – Automating deployment, scaling, and (lifecycle & resource) management of containerized applications

• Current solutions & limitations
  – NVidia-Docker: expose GPU devices to containers
    • Dedicate GPU allocation to container
  – K8S resource limit: control memory and CPU usage
    • GPU is not manageable resource yet
  – KubeFlow: A TF-operator to deploy containerized TF job as a set of K8S applications
    • Naïve round-robin scheduling without scaling and management
Proposed Solutions: Multi-tenant GPU controller

- **Objective**
  - Treat GPU as the **first class** resource like CPU
  - Allow users to specify the **max** and **min** requirements for GPU utilization and memory usage

- **Approach**
  - Intercept CUDA driver & runtime API
  - Forward requests to a centralized scheduler for CPU and memory control
  - Similar to conVGPU, but focus more on GPU utilization control and GPU assignment
Proposed Solutions: Elastic-KubeFlow

- An enhanced K8S TF-operator over KubeFlow

<table>
<thead>
<tr>
<th></th>
<th>Auto-Deployment</th>
<th>Scheduling</th>
<th>Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeFlow</td>
<td>✓</td>
<td>Round-Robin</td>
<td></td>
</tr>
<tr>
<td>Elastic-KubeFlow</td>
<td>✓</td>
<td>Perf. Aware Placement</td>
<td>Scale-up when util. is low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scale-down when wait time is high</td>
</tr>
</tbody>
</table>

![Diagram showing job queue and system loading scenarios](image-url)
Proposed Solutions: Elastic-KubeFlow

• An enhanced K8S TF-operator over KubeFlow

<table>
<thead>
<tr>
<th></th>
<th>Auto-Deployment</th>
<th>Scheduling</th>
<th>Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeFlow</td>
<td>🟢</td>
<td>Round-Robin</td>
<td>✗</td>
</tr>
<tr>
<td>Elastic-KubeFlow</td>
<td>🟢</td>
<td>Perf. Aware Placement</td>
<td>Scale-up when util. is low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scale-down when wait time is high</td>
</tr>
</tbody>
</table>

**System in Low Loading**

- Job queue
  - Job1 worker1
  - Job1 worker2
  - Job1 worker3
  - Job1 worker4

- TF-operator

**System in High Loading**

- Job queue
  - Job1 worker1
  - Job1 worker2
  - Job1 worker3
  - Job1 worker4

- Job queue
  - Job5 worker1
  - Job6 worker1

**Total job run time:** 4:11:44 ➔ 3:16:54 (22%)

**Total job wait time:** 4:45:02 ➔ 2:57:37 (38%)
Distributed Deep Learning

- **Model Parallelism**
  - Within a node: shared memory, **auto-managed by framework**
  - Across nodes: message passing, model rewritten by **developers**

- **Data Parallelism**
  - Parameter server:
    - Asynchronous centralized comm.
      - Faster converge time, but higher network BW requirement
    - Main strategy in TF
  - All reduce:
    - Synchronous P2P comm.
      - Higher latency delay, but more balanced network traffic (avoid hotspot)
    - Recent optimized imp. by **Horovod**
Distributed Model Training

- Why distributed model training?
  - Shorter training time
  - Fully utilize computing resources

- Non-negligible overhead
- More tuning knobs: batch size, learning rate, #PS
Proposed Solutions: Elastic-TensorFlow

• Why we want to dynamically add/remove workers from a training job without checkpoint-restart?
  – Auto-tuning PS/Worker ratio at runtime
  – Reach desired performance with minimum cost
  – Maximize system utilization & throughput (Combine with our elastic-kubeflow controller)


Distributed training strategies for a computer vision deep learning algorithm on GPU cluster
AI for Systems

• **Time prediction for optimizing job execution**
  – Apply FCN, RNN for complex parallel DAG

• **Anomaly & failure prediction for minimizing cost**
  – DNN along might not be enough...
    • Using SVM for rare class classification
    • Using bayesian network or decision tree for root cause diagnosis
    • Using probability distribution for system metrics prediction

• **Auto-scaling & Scheduling for maximizing system performance**
  – Apply reinforcement learning: A3S, Deep Q-learning
Time Prediction of Hadoop Execution

\[ f(\text{job profile, resource spec, exe config}) = \text{job execution time} \]

- A parallel execution job
- Over 100 execution configurations
- Cloud platform provides varied compute instance types
- Inexperienced users for performance optimization
Time Prediction of Hadoop Execution

- **Step 1**: Job Profiling
  - Collect job features
- **Step 2**: Job classification
  - Improve prediction accuracy
- **Step 3**: Model prediction
  - Fully-Connected NN
- **Step 4**: Optimization
  - Search optimal configurations
Evaluation Results

- Workload from HiBench, a Hadoop benchmark suite

**Prediction Accuracy**

- DecisionTree: 16%
- SVM: 12%
- NN: 8%

More accurate time prediction than traditional ML methods

**Performance Improvement**

10~50% performance improvement by choosing the proper execution configurations
Time Prediction of Hive Query

- Hive: A query engine on Hadoop
  – Complex workflow represented by DAG

```sql
SELECT tmpl.key, count(*)
FROM t1
JOIN /*JOIN1*/ (SELECT key, avg(value) AS avg
    FROM t1
    GROUP BY /*AGG1*/ key) tmpl ON (t1.key = tmpl.key)
JOIN /*JOIN1*/ t2 ON (tmpl.key = t2.key)
WHERE t2.value > tmpl.avg
GROUP BY /*AGG2*/ t1.key;
```
Time Prediction of Hive Query

- **RNN model**
  - Serialized DAG workflow with arbitrary job sequence length
  - Stored state for capturing job dependency effects
- **Two level prediction & optimization**
  - Query level (Hive) and job level (Hadoop)
Evaluation Results

• Workload from TPC-H benchmarks

Prediction Accuracy

RNN has the lowest error rate comparing to DNN and other methods

Performance Improvement

Improve performance by over 50% when both Hadoop and Hive configurations are optimized
Thanks