Towards Predicting Performance of GPU-dependent Applications on the Example of Machine Learning in Enterprise Applications

Symposium on Software Performance 2017 (SSP 2017)

Felix Willnecker, Helmut Krcmar

fortiss GmbH
An-Institut Technische Universität München
Agenda

• Introduction
• Example Project
• Problem Domain #1 Model Complexity
• Problem Domain #2 GPU Representation
• Problem Domain #3 Memory Bandwidth
• Problem Domain #4 Monitoring
• Conclusion
Introduction

Motivation

• Graphics Processing Units (GPU) become integral part of Enterprise Applications [Krueger2011] driven by
  – Video- and Image processing
  – IoT/Industry 4.0, especially Predictive Maintenance
  – Blockchain algorithms
  – Artificial Intelligence and Machine Learning in general [Janakiram2017]
• GPUs increase their processing capabilities faster than CPUs these days
  – “Moore's Law is dead, GPUs will soon replace CPUs” - Jensen Huang, CEO NVIDIA
• Adoption in terms of supporting frameworks¹ and infrastructure providers²
• Known problem in the domain of High Performance Computing [Kothapalli2009]
• Architecture-level Performance Models do not intentionally support GPU as a dedicated Resource

¹ Tensorflow GPU support: https://www.tensorflow.org/tutorials/using_gpu
² AWS EC2 GPU support: https://aws.amazon.com/de/ec2/elastic-gpus/
Introduction

Use Cases

- Supporting and optimizing architectural design decisions:
  - Which framework to use?
  - Implementing CPU algorithm or GPU algorithm?
  - Ideal interface to handover tasks to GPU (optimal usage of memory bandwidth and limited CPU)

- (Performance) Model-based Capacity Planning
  - Selection of fitting instances more complex due to additional resource
  - Number of required instances, up/down scaling
  - Cost management and optimization

- Autonomous systems adapting to workload
  - Scaling based on load on additional resource
  - CPU vs GPU decisions during runtime
  - Provisioning of GPU instance only when required (cause they are expensive)
**Example Project**

Self-learning Smart Software Project and Team Matching

**Human Resources**
- Successful software projects depend heavily on the people doing these projects (Coudert, T./2012)
- Software projects often "take what they can get"; as choice is often limited (Kessler, R.; et al/2012)
- Project partner chose another approach: large pool of talents (students, employees, and freelancer)
- Teams are matched according to skill and previous project data manually

**Sales**
- Sales pipeline for software projects outworn recruitment in some areas or for some skills
- Sales works on „we have done something similar“ basis (Loth, R; et al/2010)
- They disregard talent and skill availability
- Opportunity scanning from a fulfillment perspective is not conducted yet (Misirli, A; Bener, A/2014)
Example Project
Self-learning Smart Software Project and Team Matching

• (Self-learning) team matching algorithm
• Skill Matrix:
  – Every talent has a number of skills
  – Each skill has a number of data points (interview, personal assessment, coding tests, github, project performance, …)

• Task: How to weight data points rating the skills?

• Input: (Relevant) Skill Matrix
• Classes: Project success (money and customer satisfaction)
• Primary output: Weight of data point for relevant skills
• Secondary output: Predict performance using PCM as pilot study for opportunity scanner
Problem Domain #1 Model Complexity

Simple Neuronal Network Processing

- CPU spawns task and transfers Skill Matrix and classifier in GPU VRAM
- GPU derives and tests model
- Massive parallel execution (parallel execution of a skill)
- Results assembled and processes from CPU
Problem Domain #1 Model Complexity
Simple Neuronal Network Processing

- Simple representation of this model in PCM
- Ignore CPU and treat Processing Resource as GPU, including many replicas
Problem Domain #1 Model Complexity

Simple Neuronal Network Processing

- **SimuCom** ➔ Generated class exceeds 65k
- **Eventsim** ➔ StackOverflow
- **SimuLizar** ➔ Simulation never started
- Other issues:
  - Eclipse froze multiple times
  - Loading the editor takes endlessly
  - Maintainability of the project

- **Ideas to reduce complexity**
  - Replace Forked Behaviors by External Async Calls (solves SimuCom issue)
  - Add additional element similar to Loop Action that spawns an amount of these actions
Problem Domain #2 GPU Representation

- No dedicated GPU representation available in PCM
- VRAM of Graphics Card is often a bottleneck and needs a representation as well
Problem Domain #3 Memory Bandwidth

- Memory bandwidth is one of the main bottlenecks in GPU processing today
- Potentially large impact on performance
- Simple approach:
Problem Domain #3 Memory Bandwidth

- Alternative approach:
  - Internal Linking Resource connecting Processing resources with Memory resources
  - Simulate bandwidth and latency when accessing these resources
  - Allow to transfer data internally instead of only externally
Problem Domain #4 Monitoring

- New resources require monitoring solution that allows to observe their behavior
- Closed source and hardware dependent solutions available today (e.g. AMD System monitor)
- Integrated monitoring solution only on system level
- Opportunity for (open-source) APM solutions
Conclusion

- Architecture-level performance models not ready for the GPU age
- Small fixes allow to get simulations up and running
- Mature solutions for the integration still require a lot of work

- Problem Domain #1 Model Complexity
  - 30%
- Problem Domain #2 GPU Representation
  - 60%
- Problem Domain #3 Memory Bandwidth
  - 20%
- Problem Domain #4 Monitoring
  - 0%
References

• **J. Krueger et al.**, Applicability of GPU Computing for Efficient Merge in In-Memory Databases." In: ADMS@VLDB. 2011, pp. 19-26.

• **Janakiram MSV**, In The Era Of Artificial Intelligence, GPUs Are The New CPUs, Forbes, Aug 2017


References


Felix Willnecker
Helmut Krcmar

Performance Management Group
fortiss GmbH
An-Institut Technische Universität München
Guerickestraße 25 · 80805 München · Germany

tel +49 89 3603522 512    fax +49 89 3603522 50

Felix.Willnecker@fortiss.org
www.fortiss.org