Reciprocity between HPC and Deep Learning

Xipeng Shen

Computer Science Department
North Carolina State University
About Me

• Programming Systems and Machine Learning
  • Making computing more **intelligent** and **efficient**
Graduated

- 8 Ph.Ds
  - 4 assistant professors
    - Rutgers University (Zheng Zhang 2012)
    - UC Santa Barbara (Yufei Ding 2017)
    - UC Riverside (Zhijia Zhao 2015)
    - Colorado School of Mines (Bo Wu 2014)
  - 4 industry
    - Microsoft, Google, IBM, Qualcomm
- 3 recent masters
  - Google, Amazon, Qualcomm
Current Sponsors

- NSF
- Department of Energy
- LLNL
- Oak Ridge National Laboratory
- Google
- Cisco
- IBM
Programming systems and Intelligent Computing for future
This Talk

HPC

Reciprocity

Deep Learning
Generalized Strength Reduction for Enabling Algorithmic Optimizations
Motivation

Computing efficiency is a key.

sources: IBM
Strength Reduction

- A basic compiler concept

\[
\frac{b}{2} \rightarrow b \gg 1
\]

Traditional: only instruction level.

Our Goal:
Generalize it with Triangle Inequality
Triangle Inequality

\[ |a-b| \leq d \leq a+b \]
Example

Triangular Inequality:
\[ a-b \leq d \leq a+b \]

K-Means

#Distances: \( O(K) \) V.S. \( O(N \times K) \)
TI Optimization V.S. Strength Reduction

• Connection:
  – Replacing expensive distance computations with cheaper bounds for comparisons.

• Challenges:
  – Minimize costs of TI usage while avoiding calculations.
  – Extend to other algorithms.
K-Means (K=1024)

Clustering results are same as original method’s.

Baseline: Classic K-means
(16GB, 8-core Intel Ivy Bridge)

Yinyang K-Means

Code link in ICML’15 paper.
On K-Means

Baseline: Classic K-means (16GB, 8-core)

UKbench Speedup

- Elkan's
- Drake's
- TOP

Speedup (X)

K

0  4  64  1024  10000

XX
Average speedups: 50X vs 20X. Save at least 93% calculations.

Intel i5-4570 CPU and 8G memory
Angular Triangle Inequality (ATI)

\[ 130^\circ \geq \theta_{qt} \geq 110^\circ \]
\[ \cos 110^\circ \cdot |\vec{q}| \cdot |\vec{t}| \geq \vec{q} \cdot \vec{t} \geq \cos 130^\circ \cdot |\vec{q}| \cdot |\vec{t}| \]

Holds in spaces of any (>1) dimensions!
ATI always gives tighter bounds than TI does!

(detailed proof in PLDI’17 paper).
Deep neural network
Speedup (X) on Training

Datasets

Mnist  f-Mnist  Cal01  Newsgroup  Micro-norb

GTX980

TitanX

(Baseline: CuBlas)
Speedup (X) on Inference on Tablet (Nexus 7 NVIDIA Tegra 3 T30L)
DNN for Sparse Format Selection

Deep Learning -> HPC
Problem

SpMV: core of many HPC applications.

Sparse Matrix

\[
\begin{pmatrix}
1.0 & 0 & 5.0 & 0 & 0 & 0 & 0 \\
0 & 3.0 & 0 & 0 & 0 & 0 & 11.0 \\
0 & 0 & 0 & 9.0 & 0 & 0 & 0 \\
0 & 0 & 6.0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 7.0 & 0 & 0 & 0 \\
2.0 & 0 & 0 & 0 & 10.0 & 0 & 0 \\
0 & 0 & 0 & 8.0 & 0 & 0 & 0 \\
0 & 4.0 & 0 & 0 & 0 & 0 & 12.0
\end{pmatrix}
\]

Storage formats

- CSR
- COO
- DIA
- ELL
- HYB
- ...

Multi-fold performance differences.

No one fits all.
Basic Idea

Treat matrix as an image, use *image recognition* methods for selection.
Special Challenge I

- Input representation: fixed size required.
- Image scaling does not work well

(a) An example original matrix. (b) Normalized binary matrix.

Solution: Binary matrix, density matrix, histogram matrix.
Special Challenge II

- DNN structure design
  - Early Merging versus Late Merging
Special Challenge III

- Architecture Sensitivity
  - Best formats differ on different machines for a matrix
Compared to Prior DT-based Method

Accuracy: 85% boosted to 93%. (9200 matrices; Formats: coo, csr, dia, ell)

PLDI’13
Benefits from Transfer Learning

From Xeon E5-4603 to Radeon A8-7600
Other Work: Egeria [SC’17]

A synthesizer of HPC advising tools.

- HPC Documents → Advising Sentence Recognition → Collection of advising sentences → Relevance Calculation → Relevant advising sentences

Diagram:

- User
- Query
- Advising Sentence Recognition
- Collection of advising sentences
- Relevance Calculation
- Relevant advising sentences

Text:

A synthesizer of HPC advising tools.

5. Performance Guidelines

5.2. Maximize Utilization

5.2.3. Multiprocessor Level

- At an even lower level, the application should maximize parallel execution between the various functional units within a multiprocessor.
- As described in Hardware Multithreading, a GPU multiprocessor relies on thread-level parallelism to maximize utilization of its functional units.
- The number of clock cycles it takes for a warp to be ready to execute its next instruction is called the latency, and full utilization is achieved when all warp schedulers always have some instruction to issue for some warp at every clock cycle during that latency period, or in other words, when latency is completely "hidden".
- Experimentation is therefore recommended.
- Applications can also parameterize execution configurations based on register file size and shared memory size, which depends on the compute capability of the device, as well as on the number of multiprocessors and memory bandwidth of the device, all of which can be queried using the runtime (see reference manual).
- The number of threads per block should be chosen as a multiple of the warp size to avoid wasting computing resources with under-populated warps as much as possible.
Other Undergoing Efforts

• Emerging memory technology
  • w/ Intel; NSF award. [Micro’17]

• GPU program optimizations
  • Google Faculty Award; DOE Career Award, LLNL award. [ASPLOS’11,Micro’14,Micro’15,Micro’17]

• Speeding up DNN ensemble training
  • ORNL award. [Micro’17]

• Machine learning fast hyperparameter tuning
  • ORNL award.
To Learn More

PICTure Group

Programming systems and Intelligent Computing for future

https://research.csc.ncsu.edu/picture/

Or google “Xipeng Shen”