GPU Acceleration of Fuzzy Tags

Natalie Jadue and Michael Kaminski

JOBVITE

PURDUE UNIVERSITY

PROJECT GOAL

Improving Computational Efficiency

- Jobvite runs Fuzzy Tags computations on the cloud
  - Amazon Web Services (AWS) charges *hourly* rates for cloud computing
  - Faster computation will yield immediate cost savings
- Goal: Reduce computation time
  - We are NOT looking to *use* Fuzzy Tags, just *improve* its implementation
- Strategy: Re-write computationally expensive code to utilize GPU acceleration

BACKGROUND

What is Fuzzy Tags?

- Model used by Jobvite for ranking job applicants
  - Purpose: Determine the *likelihood* of a candidate possessing a certain skill
  - "Tags" refers to labels on documents
  - "Fuzzy" refers to a probabilistic application of these labels
- Skills focused
  - Can help reduce bias in the hiring process
    - Model doesn't use gender, age, ethnicity, or other traits that could indicate these

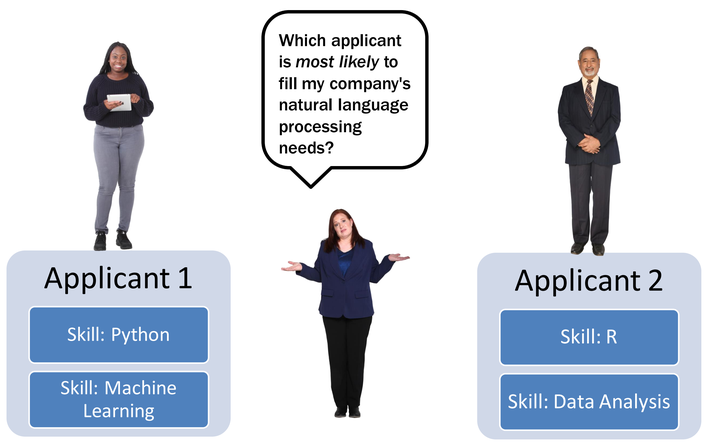

Figure 1: Example use case of Fuzzy Tags

CHALLENGE TO OVERCOME

Computation and Communication Runtime Trade-Off

- Computation on the CPU is slow
  - No need to transfer information—all data remains in CPU memory
- Computation on the GPU is fast
  - Data transfers between CPU and GPU memory can be slow
- Challenge:
  - Minimize data transfers between CPU memory and GPU memory
  - Overlap data transfers with GPU computations to "hide" the transfer time

METHODOLOGY

Addressing the Challenge

1. Tool Identification:
   - Determined TensorFlow does not provide adequate control over data locality
   - Identified CuPy as providing control over data transfers between CPU and GPU
2. Initial Benchmark Tests:
   - Implemented simplified Fuzzy Tags computations in CuPy
   - Produced benchmark test showing GPU accelerated computation in CuPy to be 130-140x faster than CPU computations
3. Creating Fuzzy Tags GPU Implementation:
   - Rewrote portions of Jobvite's Fuzzy Tags code to use CuPy
   - Wrote unit tests to ensure the GPU implementation's functionality and correctness
4. Testing GPU Implementation:
   - Benchmarked runtimes in a mock use case by using Fuzzy Tags as a search engine for Stack Overflow questions

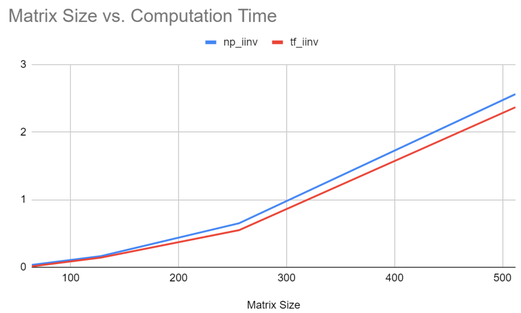

Figure 2: TensorFlow's lackluster performance in comparison to NumPy

TOOLS

Why CuPy?

- Python library aiming to create a GPU accelerated NumPy
  - Most functions synonymous with NumPy
  - Easy to convert the Fuzzy Tags NumPy CPU implementation into CuPy
- Supports CUDA Unified Memory
  - Memory manager handles data transfers via a page fault system
  - No need for hard coded memory copies in the naïve implementation
  - Frequently used data remains resident on the GPU
- Supports manual data transfers between CPU and GPU
  - Enables fine-tuning performance achieved by CUDA Unified Memory

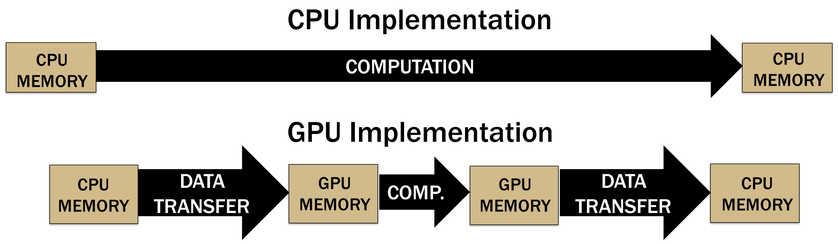

Figure 3: Flow of CPU and GPU programs

CONCLUSIONS AND FUTURE DIRECTIONS

Results

- Created a working version of Fuzzy Tags running computations on the GPU using CuPy and CUDA Unified Memory
  - Implemented additional optimizations to the GPU accelerated version
- Compared runtime of original CPU NumPy implementation with the GPU CuPy implementations
  - Used Fuzzy Tags as a search engine for questions from stackoverflow.com
    - Jobvite data is private, Stack Overflow data is public but similar in structure
  - Read in search queries and returned the question most likely to match the search
- The GPU version is more than 3.5x faster in the Stack Overflow benchmark
  - At this performance level, the speed of computations *do not* cover the additional hardware costs for AWS instances with a GPU
- Better performance expected when run on Jobvite-owned data
  - Larger computations and additional data structure will help the GPU performance

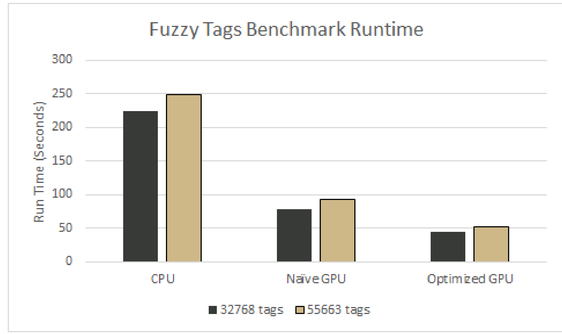

Figure 4: Runtime vs. Implementation type

Future Work

- Implement portions of Fuzzy Tags in Numba
  - Numba is a "just-in-time" compiler which compiles slow Python code into fast C code at runtime
- Incorporate Basic Linear Algebra Subprograms (BLAS) to make code more efficient
  - Specialized implementations of basic operations taking advantage of hardware specifics to improve performance
- Investigate optimal sorting of Fuzzy Tags input data
  - Grouping pieces of data which are commonly used in tandem can reduce total data transfers between the CPU and GPU

ACKNOWLEDGEMENTS AND REFERENCES

Acknowledgements

We would first like to thank the Data Mine team, Dr. Mark Ward and Maggie Betz, for being supportive throughout the semester and for their continued efforts to help us succeed. We would also like to thank everyone from Jobvite for providing us guidance and supplying the project. Lastly, thank you to Jon Roose who was fundamental in mentoring us this year. We would not have been able to finish the project without his advice and expertise.

References

- TensorFlow: https://www.tensorflow.org/
- CuPy: https://cupy.dev/
- NVIDIA CUDA Documentation: https://docs.nvidia.com/cuda/
- Jobvite: https://www.jobvite.com/