FULL-W2V: Fully Exploiting Data Reuse for Word2Vec on GPU-Accelerated Systems

Thomas Randall, Tyler Allen, Rong Ge
{tlranda, tnallen, rge}@clemson.edu

ICS ’21, Virtual Event; June 17, 2021

Funded in part by NSF Grants: CCF-1551511 and CNS-1551262
GPU W2V Not Faster than CPU

- 3-layer ANN
  - Words \( w \rightarrow d \)-dimensional embeddings \( e \)
  - Prior ports based on data-intensive implementation
  - Suboptimal usage of GPU memory hierarchy

Sentence Contents (“Context Windows”)
- Context windows include adjacent words

Noise Contrastive Samples (“Negatives”)
- \( n_0 \): put
- \( n_1 \): pastel
- \( n_2 \): rehearsal

Input          Projection          Output
\( w_i \)       \( e_i \)          \( e_{i+2} \)
\( e_{i+1} \)
\( e_{i-1} \)
\( e_{i-2} \)
GPU W2V Not Faster than CPU

- 3-layer ANN
  - Words $w \rightarrow d$-dimensional embeddings $e$
  - Prior ports based on data-intensive implementation
    - Suboptimal usage of GPU memory hierarchy

---

**Input**

- $w_i$

**Projection**

- $e_i$
- $e_{i+1}$
- $e_{i-1}$
- $e_{i-2}$

**Output**

---

Sentence Contents ("Context Windows")

- Context windows include adjacent words

Noise Contrastive Samples ("Negatives")

- $n_0$
- $n_1$
- $n_2$

- put
- pastel
- rehearsal

---

- 3-layer ANN
- Words $w \rightarrow d$-dimensional embeddings $e$
- Prior ports based on data-intensive implementation
- Suboptimal usage of GPU memory hierarchy
GPU W2V Not Faster than CPU

- 3-layer ANN
  - Words $w \rightarrow d$-dimensional embeddings $e$
  - Prior ports based on data-intensive implementation
  - Suboptimal usage of GPU memory hierarchy

Sentence Contents (“Context Windows”)

Noise Contrastive Samples (“Negatives”)

- Introduction
- Problem
- Techniques
- Results
- Conclusions
GPU W2V Not Faster than CPU

- 3-layer ANN
  - Words $w \rightarrow d$-dimensional embeddings $e$
- Prior ports based on data-intensive implementation
- Suboptimal usage of GPU memory hierarchy
- FULL-W2V reduces access and improves locality
  - Leverage memory hierarchy based on algorithm’s access pattern

Sentence Contents ("Context Windows")
- Context windows
- Include adjacent words

Noise Contrastive Samples ("Negatives")
- $n_3$
- $n_4$
- $n_5$
- agree
- ash
- short
Negative Sample Reuse: Register-W2V

- Challenge: Negatives are random and have lower reuse
Negative Sample Reuse: Register-W2V

- Challenge: Negatives are **random** and have **lower reuse**
- Opportunity: Operations have **independent order**
Negative Sample Reuse: Register-W2V

- Challenge: Negatives are random and have lower reuse
- Opportunity: Operations have independent order

![Diagram](image-url)
Negative Sample Reuse: Register-W2V

- Challenge: Negatives are random and have lower reuse
- Opportunity: Operations have independent order
- Solution: Use registers for maximum reusability
  - Minimize up-front memory latency, maintain locality
  - Improved pipeline utilization
  - Maintain scheduling flexibility, reduce stress for Shared Memory

\[
\begin{align*}
  i-2 & : \text{Context} \\
  i-1 & : \text{windows} \\
  i+1 & : \text{adjacent} \\
  i+2 & : \text{words} \\
\end{align*}
\]
Negative Sample Reuse: Register-W2V

- Challenge: Negatives are random and have lower reuse
- Opportunity: Operations have independent order
- Solution: Use registers for maximum reusability
  - Minimize up-front memory latency, maintain locality
  - Improved pipeline utilization
  - Maintain scheduling flexibility, reduce stress for Shared Memory
Context Word Reuse: FULL-W2V

• Different Pattern: Context Words have more reuse
Context Word Reuse: FULL-W2V

- Different Pattern: Context Words have more reuse
Context Word Reuse: FULL-W2V

- Different Pattern: Context Words have more reuse
- Allocation: Shared Memory leverages longer-term reuse
  - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer
Context Word Reuse: FULL-W2V

- Different Pattern: Context Words have **more reuse**
- Allocation: Shared Memory leverages longer-term reuse
  - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer

 Buffer:

```
 Context   windows   include   adjacent
```

\[
i-1 \quad i \quad i+1 \quad i+2
\]

\[
\text{Context windows include adjacent words put pastel rehearsal}
\]
Context Word Reuse: FULL-W2V

- Different Pattern: Context Words have more reuse
- Allocation: Shared Memory leverages longer-term reuse
  - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer

Buffer:

<table>
<thead>
<tr>
<th>Context</th>
<th>windows</th>
<th>include</th>
<th>adjacent</th>
<th>words</th>
</tr>
</thead>
</table>

\[
i-2 \quad i-1 \quad i \quad i+1 \quad i+2
\]

Context windows include adjacent words agree ash short
Context Word Reuse: FULL-W2V

- Different Pattern: Context Words have more reuse
- Allocation: Shared Memory leverages longer-term reuse
  - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer
Context Word Reuse: FULL-W2V

- Different Pattern: Context Words have more reuse
- Allocation: Shared Memory leverages longer-term reuse
  - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer

Buffer:

<table>
<thead>
<tr>
<th>Context</th>
<th>windows</th>
<th>include</th>
<th>adjacent</th>
<th>words</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

i-2  i-1  i  i+1  n_6  n_7  n_8
Context Word Reuse: FULL-W2V

• Different Pattern: Context Words have more reuse
• Allocation: Shared Memory leverages longer-term reuse
  • High performance; Explicit control; Flexible scheduling
• Management: Ring buffer

Buffer:
Resulting Memory Demand Reduction

- Demand represented as \textit{access size X (iterations)}
- Prior works cache everything but only for one window
- FULL-W2V leverages prior windows, degree of reuse
Batching best-suited for CPU
- I/O, large data structures, randomness
- Parallelism best-suited for GPU
  - Map problem parallelism to architecture features

### Coordination & Parallel Hierarchy

<table>
<thead>
<tr>
<th>Batch $i-1$</th>
<th>Batch $i$</th>
<th>Batch $i+1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sentence (grid)</strong></td>
<td><strong>Word Pair (thread block)</strong></td>
<td><strong>context word $w_{i+2}$</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>negative sample $n_1$</td>
</tr>
</tbody>
</table>

GPU trains batches

CPU Thread samples batch

- sent 0
- ns (0,1)
- ...
- ns (0,5)
- sent S
- ns (S,1)
- ...
- ns (S,5)

CUDA Stream Synchronize

NVIDIA Hyper-Q

Other Threads

Batch (stream)
**Evaluation & Platform**

- Evaluate across 3 hardware setups
- Datasets are well-established in Word2Vec literature
- Metrics & Comparisons
  - Embedding quality, overall speedup
  - Performance counters
  - GPU: Wombat, accSGNS
  - CPU: pSGNScc, pWord2Vec

<table>
<thead>
<tr>
<th>Hardware</th>
<th>GPU Specs</th>
<th>CPU Specs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylake V100</td>
<td>80 SMs 14 TFLOP/s 900 GB/s HBM2</td>
<td>Xeon Gold 6148 2 20-core CPUs 2.40 GHz</td>
</tr>
<tr>
<td>Haswell Titan XP</td>
<td>60 SMs 12.15 TFLOP/s 548 GB/s GDDR5x</td>
<td>Xeon #5-2670 v3 2 12-core CPUs 2.30 GHz</td>
</tr>
<tr>
<td>Broadwell P100</td>
<td>56 SMs 9.3 TFLOP/s 549 GB/s HBM2</td>
<td>Xeon E5-2680 v4 2 14-core CPUs 2.40 GHz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Vocabulary</th>
<th>Words/Epoch</th>
<th>Sentences</th>
<th>Words/Epoch</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text8</td>
<td>71,291</td>
<td>16,718,845</td>
<td>17,006</td>
<td>16,718,845</td>
<td>17,006</td>
</tr>
<tr>
<td>One Billion Words</td>
<td>555,514</td>
<td>804,269,957</td>
<td>30,607,795</td>
<td>804,269,957</td>
<td>30,607,795</td>
</tr>
</tbody>
</table>
### Embedding Quality

- Limited to semantically similar implementations
- FULL-W2V does not affect embedding quality
  - Differences fall within typical variance of the evaluations

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Implementation</th>
<th>Similarity Evaluations</th>
<th>Analogy Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>WS-353</td>
<td>SimLex-999</td>
</tr>
<tr>
<td>CPU</td>
<td>Most Similar CPU</td>
<td>0.6070</td>
<td>0.3499</td>
</tr>
<tr>
<td>GPU</td>
<td>Prior Best GPU</td>
<td>0.5952</td>
<td>0.3596</td>
</tr>
<tr>
<td></td>
<td>FULL-W2V</td>
<td>0.5923</td>
<td>0.3582</td>
</tr>
</tbody>
</table>
Results

- **FULL-W2V**: Register-W2V + Context Word Reuse
  - **4.35X** total speedup previous best on V100
  - **3.85X** speedup from Register-W2V only
  - Sum data demand reduced by **91.65%**

### Demand in GB/Epoch

<table>
<thead>
<tr>
<th>Implementation</th>
<th>L1/TEX</th>
<th>L2</th>
<th>DRAM</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closest Prior</td>
<td>1,134.448</td>
<td>493.614</td>
<td>226.578</td>
<td>100.0%</td>
</tr>
<tr>
<td>Register-W2V</td>
<td>885.065</td>
<td>781.576</td>
<td>66.555</td>
<td>78.02%</td>
</tr>
<tr>
<td>FULL-W2V</td>
<td><strong>94.760</strong></td>
<td><strong>88.723</strong></td>
<td><strong>41.851</strong></td>
<td><strong>8.35%</strong></td>
</tr>
</tbody>
</table>
Deeper Analysis

- Register-W2V creates eligibility, but stalls on memory accesses
- FULL-W2V maintains eligibility, virtually eliminates scoreboard stalls

<table>
<thead>
<tr>
<th>% Active Warps</th>
<th>Prior Best GPU</th>
<th>Closest Prior Work</th>
<th>Register-W2V</th>
<th>FULL-W2V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>29.125</td>
<td>58.81</td>
<td>93.25</td>
<td>56.19</td>
</tr>
</tbody>
</table>

| Eligible Warps / Scheduler | 0.18 | 1.09 | 1.86 | **1.90** |

<table>
<thead>
<tr>
<th>Long Scoreboard Stalls</th>
<th>Register-W2V</th>
<th>FULL-W2V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11.00</td>
<td>0.97</td>
</tr>
</tbody>
</table>

| Overhead Stalls        | 7.93         | 6.35     |
Insights and Conclusion

- We present FULL-W2V
  - 4.35X prior SOTA on V100
  - 2.99X scaling from P100 to V100
- Different storage for different data
  - Register-W2V: maximize short term reuse in register
  - FULL-W2V: maximize long term reuse in shared memory
- Looking for more?
  - Our code is open source: https://github.com/tlranda/FULL-W2V
  - See the extended presentation for additional details
Acknowledgements

- Thomas Randall
  - tlranda@clemson.edu
  - tlranda.people.clemson.edu
- Tyler Allen
  - tnallen@clemson.edu
  - tnallen.people.clemson.edu
- Rong Ge
  - rge@clemson.edu
  - people.cs.clemson.edu/~rge/
- Support from NSF Grants CCF-1551511 and CNS-1551262