Persistent RNNs

(stashing recurrent weights on-chip)

Gregory Diamos

Baidu SVAIL

April 7, 2016

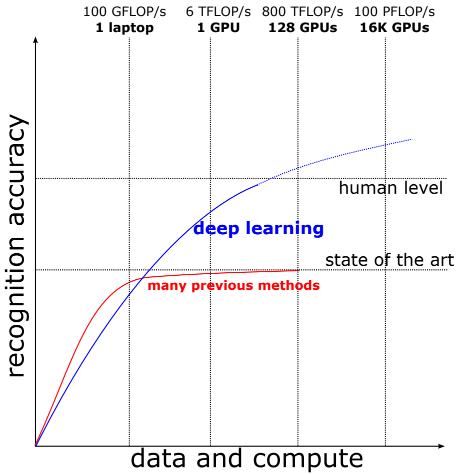
SVAIL

Think hard AI.

Goal

- Develop hard AI technologies that impact 100 million users.

Deep Learning at SVAIL

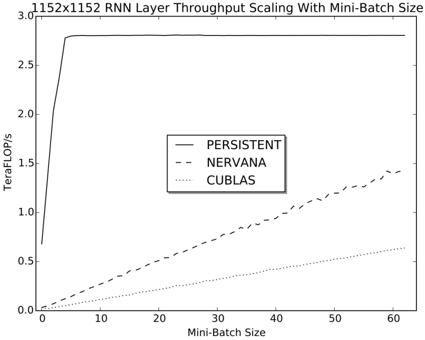


Hypothesis: deep learning scales with data and **compute**.

- Can we strong scale deep learning to the limits of technology?

Persistent RNNs

30x speedup at a mini-batch size of 4

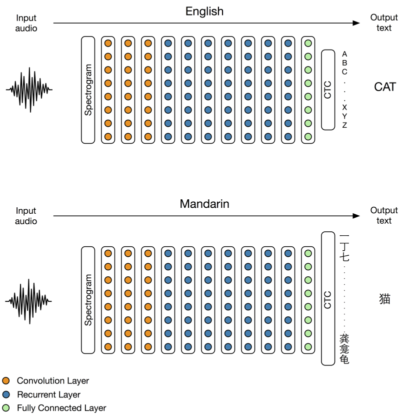


Why is reducing the mini-batch size important?

- **Train bigger and deeper models.**
- **Strong scale to more GPUs.**
- Improve efficiency of deployed models.

Training Deep RNNs

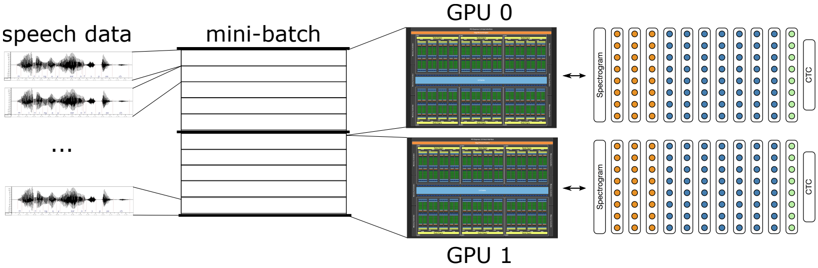
Deep speech



Near human level speech recognition in Mandarin and English

- Trained on over 10,000 hours (about 1 year) of speech data.
- 20 ExaFLOPs of work to train (7 days on 16 GPUs at 40% of peak).

Data parallel training

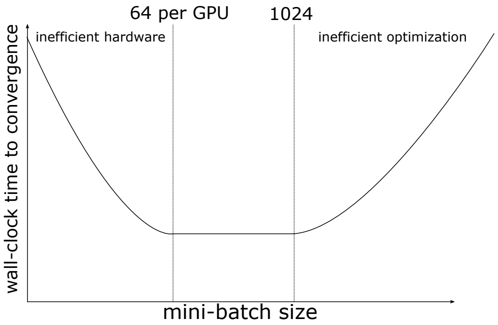


Data parallelism:

- The training data is grouped into mini-batches.
- Each GPU trains a copy of the model on a slice of the mini-batch.
- GPUs synchronize their models after a fixed number of steps.

Mini-batch constraints

So how should you choose the mini-batch size?

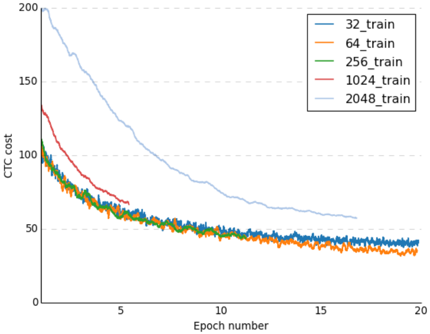


- Hardware efficiency will set a lower bound.
- Optimization efficiency will set an upper bound.

Shrinking the mini-batch per GPU enables the use of more GPUs.

Determining the batch size

The upper bound can be found empirically.

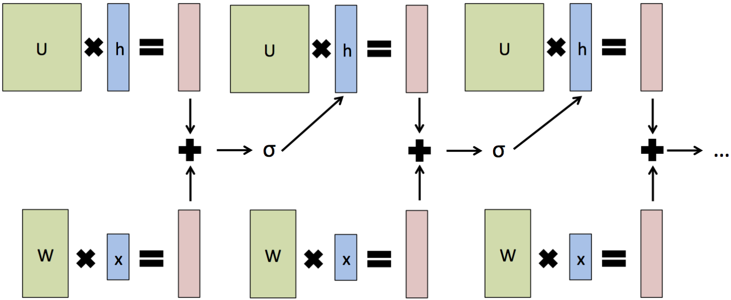


In general a hyperparameter search is needed, but a useful heuristic is:

- $momentum = 1.0 - \frac{miniBatchSize}{windowSize}$
- $learningRate = stepSize * (1.0 - momentum) * miniBatchSize$

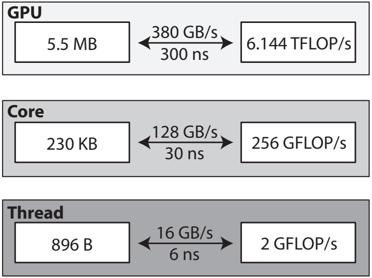
Persistent RNN Details

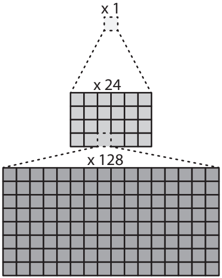
RNN primer



- RNNs built on GEMM calls reload the weights (U) each timestep.
  - However, the weights are constant, and this is wasteful.

Caching weights in registers



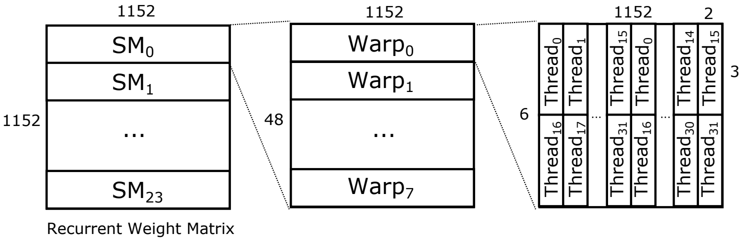


Off-chip memory is much slower and less efficient than registers.

- GPUs have more on-chip memory in registers than anywhere else.

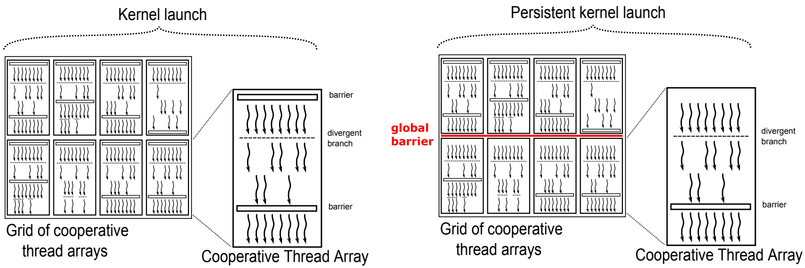
Cache RNN weights in registers and reuse them over timesteps.

Choosing the tile sizes



- Block rows avoid additional inter-CTA synchronizations.
- Each SM loads the activations into shared memory.
- Threads are interleaved to avoid shared memory bank conflicts.
- Vector loads and broadcasts amplify shared memory bandwidth.

Global barriers on GPUs



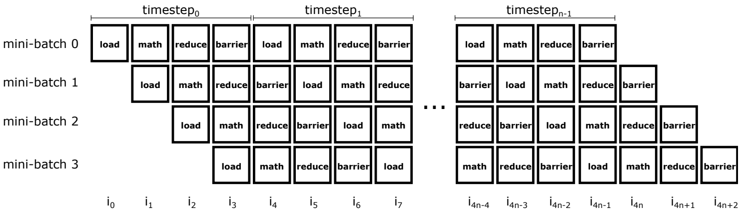
An inter-CTA barrier is implemented with a counting semaphore.

- Uses atomic, membar, and cache modified load/store operations.
- Completes in about 500ns on a TitanX GPU.

Disclaimer: global barriers violate the CUDA 7.5 model.

- CUDA does not guarantee forward progress of multiple CTAs.
- Our system implements cooperative threading for correctness.

Software pipelining



Software pipelining is used to hide latency.

- Thread local math (430ns).
- Intra-SM reduction (320ns).
- Global loads (315ns).
- Global barrier (500ns).

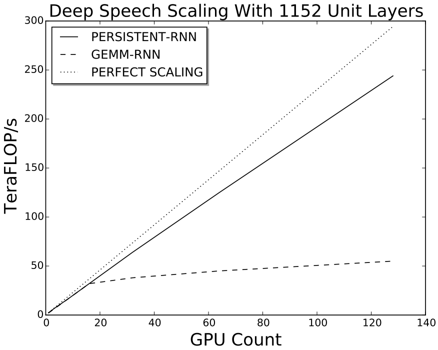
These are grouped into 4 pipeline stages, kept full with a minibatch of 4.

Strong Scaling

Scaling to 128 GPUs

Scaling results for end-to-end model training.

- 8 GPUs per node, 7GB/s infiniband between nodes.
- The algorithmic mini-batch size is fixed at 512.

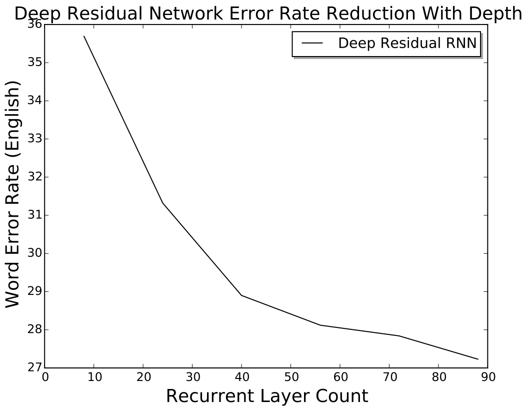


A smaller mini-batch per GPU enables the use of up to 128 GPUs.

Exploring deep residual RNNs

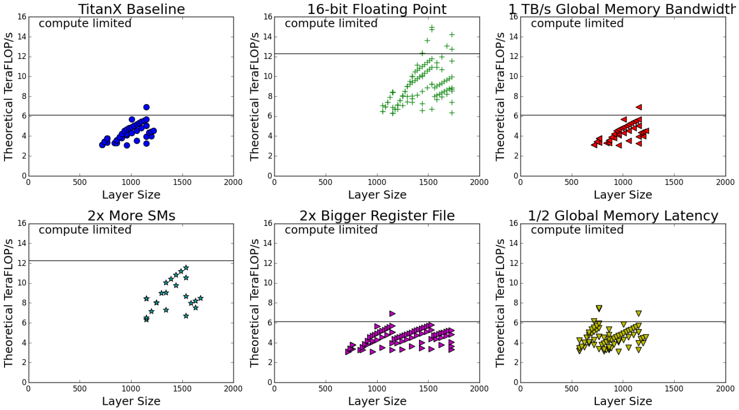
Using a mini-batch per GPU of 4 provides a 16x reduction in memory.

- Models with more parameters can now fit into GPU memory.



Results suggest that residual skip connections networks apply to RNNs.

Pascal and future

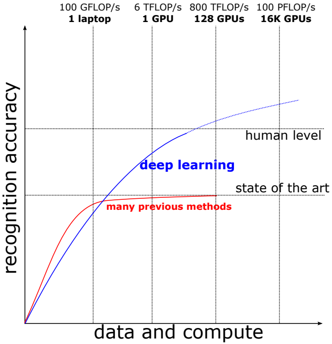


Future GPUs will enable bigger and faster RNN layers.

- bigger GPUs (more threads, more registers)
- low latency atomics between GPUs (NvLink)
- lower precision (fp16)

Conclusions

So far, deep learning for speech recognition has scaled with compute.



Persistent kernels provide a new tool for accelerating RNN training.

- Let's continue building faster computers, software, and algorithms.

What other hard AI problems will scale with deep learning and compute?

Questions

Questions?

Contact Me:

Gregory Diamos - *gregdiamos@baidu.com*

Baidu USA is hiring!

http://usa.baidu.com/careers/