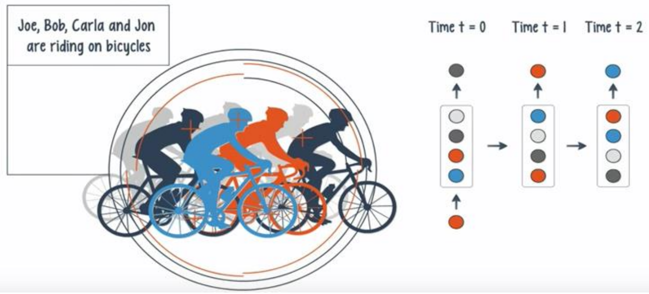
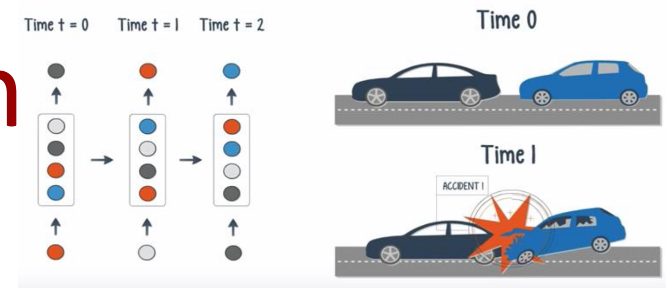
Exploring GPU Architectural Optimizations for Recurrent Neural Networks (RNNs)

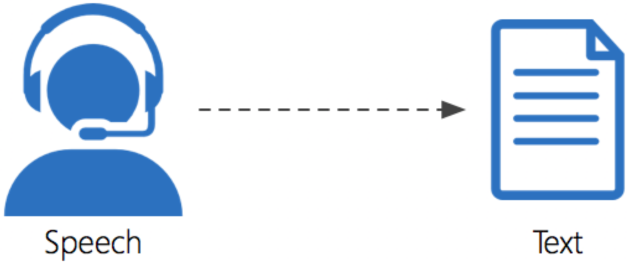
Suchita Pati
University of Wisconsin-Madison

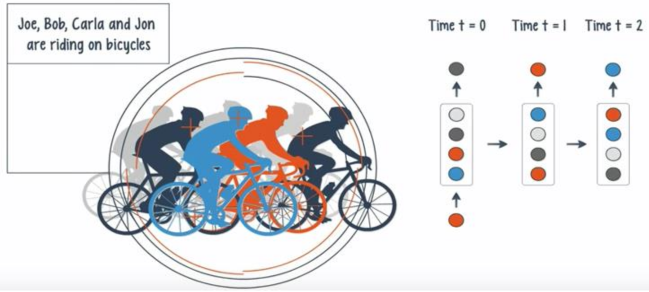


Motivation

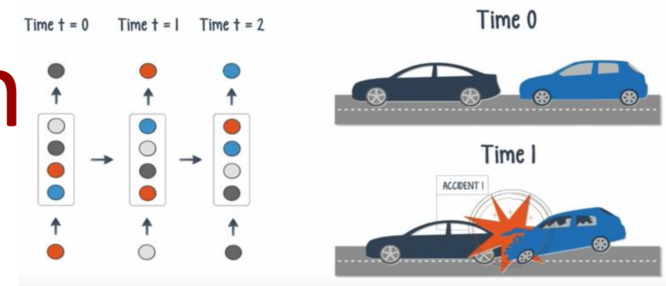


- RNNs heavily used in several important applications

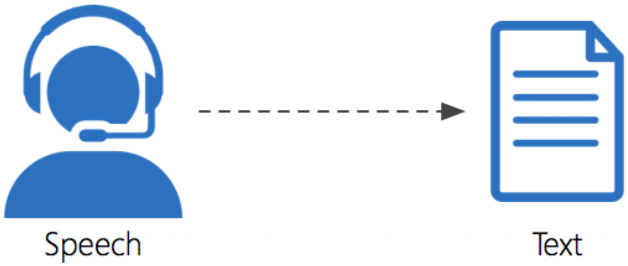


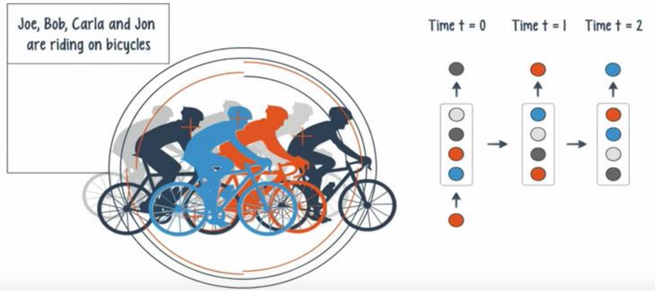


Motivation

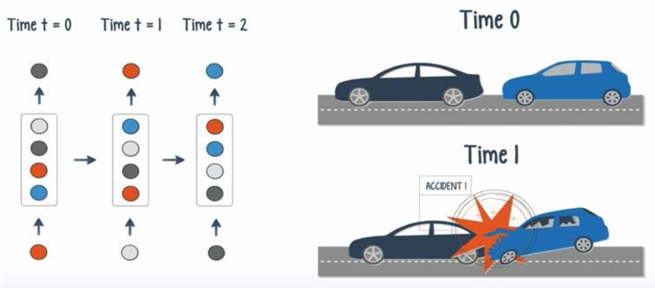


- RNNs heavily used in several important applications
- GPUs used for RNNs, but not as well studied as CNN
- RNN architecture different than CNN
- Existing CNN optimizations not very effective



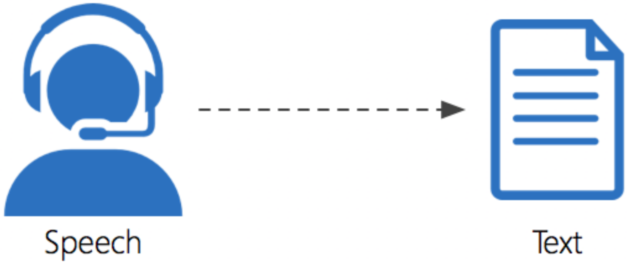


Motivation



- RNNs heavily used in several important applications

Understand RNN requirements and holistically rethink GPU arch!

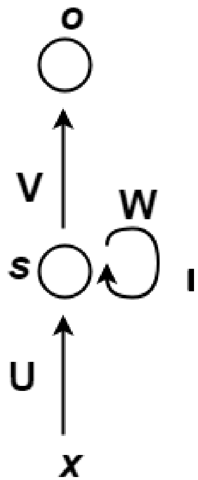


Background and Challenges

- RNNs used to recognize and predict sequences

Background and Challenges

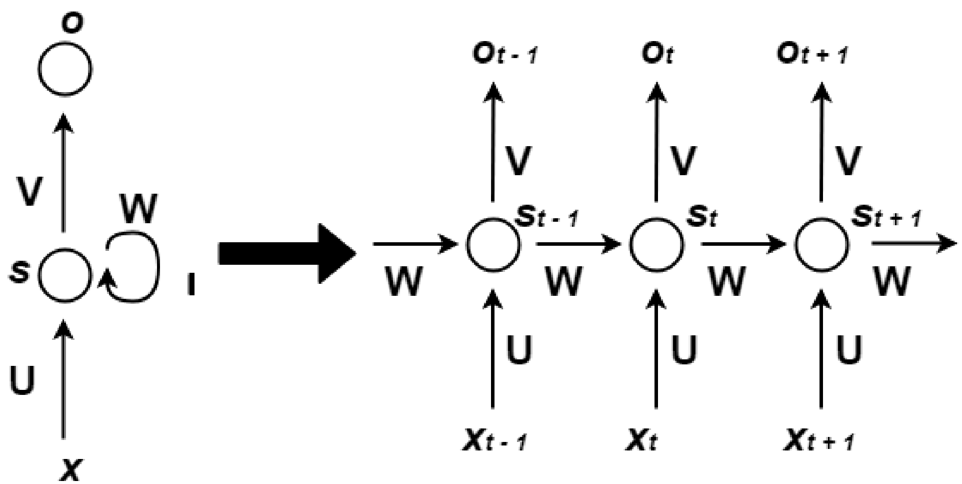
- RNNs used to recognize and predict sequences



$$\begin{aligned} x &= input \\ s &= hidden\ state \\ o &= output \end{aligned} \qquad \begin{aligned} U &= input\ weight \\ W &= recurrent\ weight \\ V &= output\ weight \end{aligned}$$

Background and Challenges

- RNNs used to recognize and predict sequences



$x = input$ $\quad U = input\,weight$

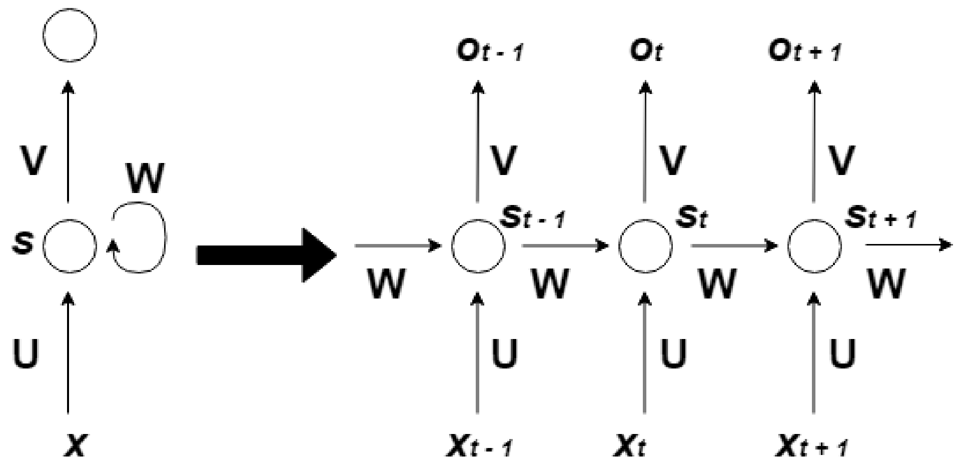
$s = hidden\,state$ $\quad W = recurrent\,weight$ $\quad t = timestep$

$o = output$ $\quad V = output\,weight$

- They need to remember previous inputs
- Have real-time deployment constraints

Background and Challenges

- Contain loops to remember information
  - sequential dependency **limits parallelism**
- Batching difficult due to strict SLA
  - poor data reuse - **high memory bandwidth**
- Read and write activations between timesteps
  - requires **high memory bandwidth**



Proposal - Compute

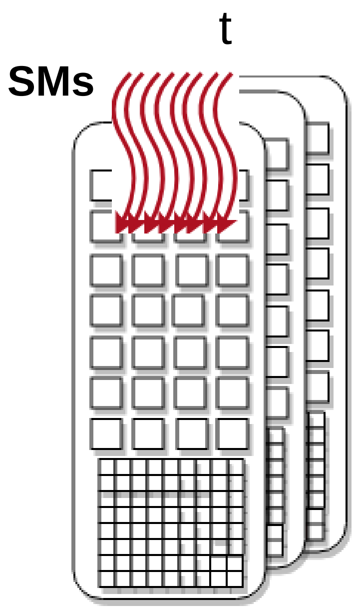
No need to wait for the entire timestep computation to finish!

Proposal - Compute

- Compute multiple timesteps in a pipelined parallel fashion

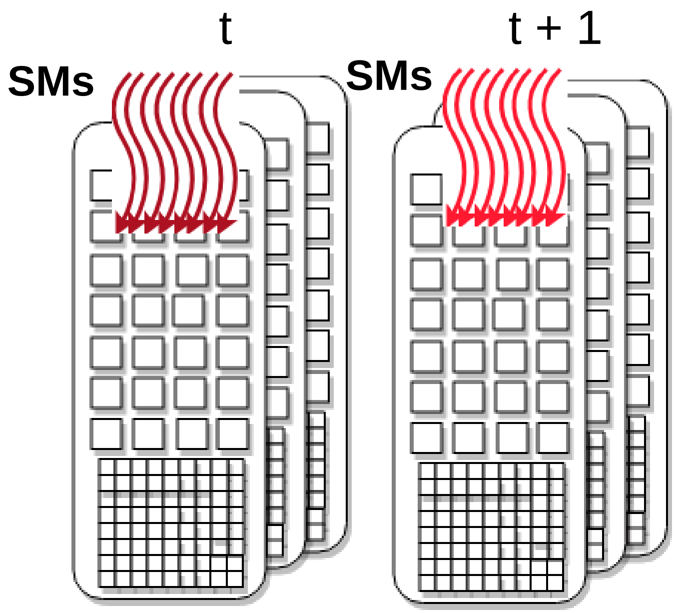
Proposal - Compute

- Map each timestep's computation to a different (set of) Streaming Multiprocessors (SM)



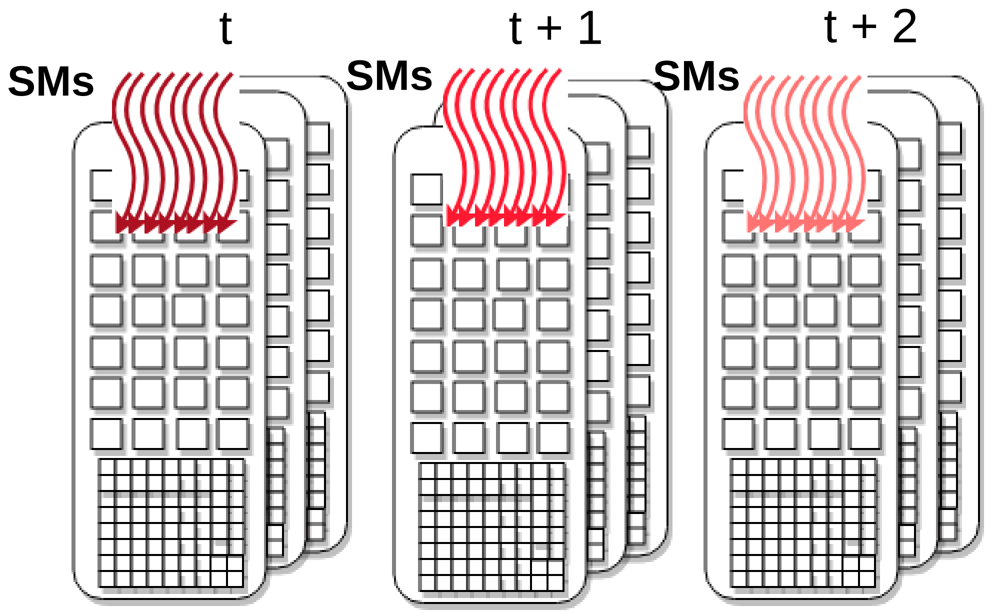
Proposal - Compute

- Map each timestep's computation to a different (set of) Streaming Multiprocessors (SM)



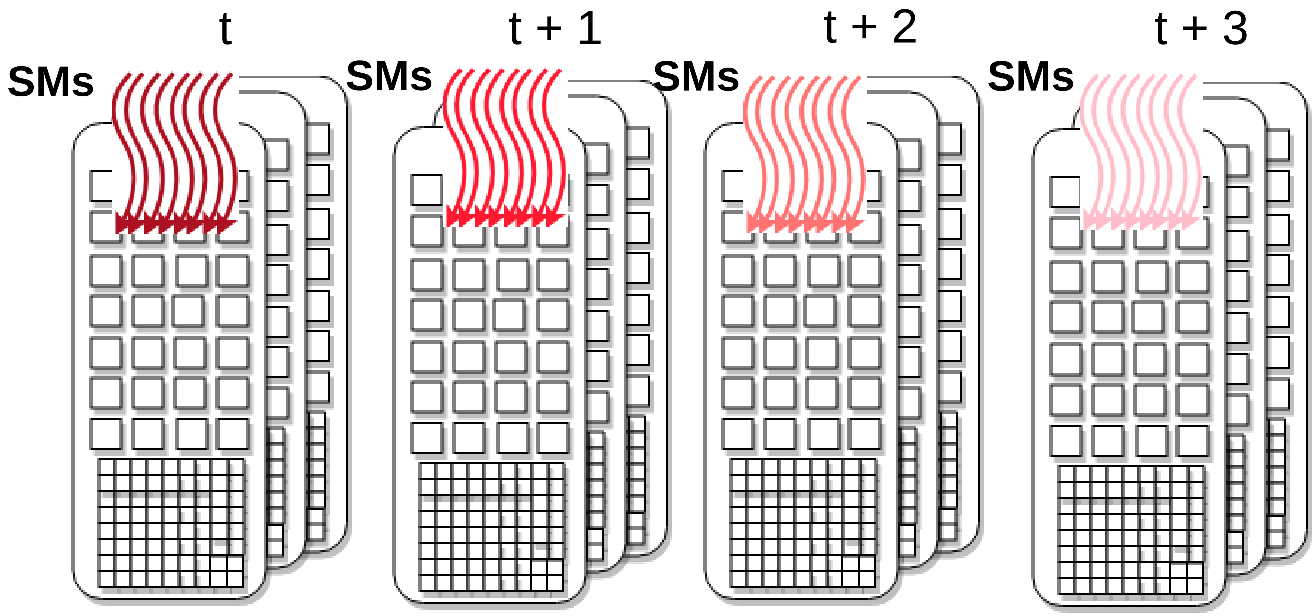
Proposal - Compute

- Map each timestep's computation to a different (set of) Streaming Multiprocessors (SM)



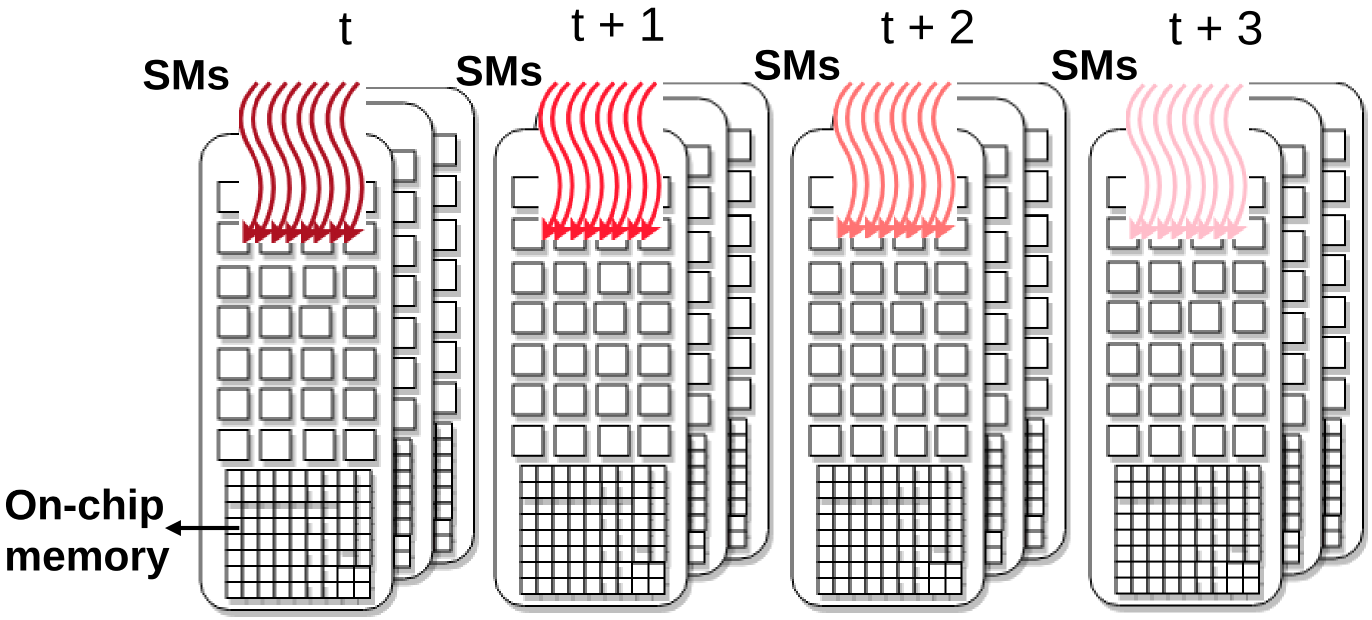
Proposal - Compute

- Map each timestep's computation to a different (set of) Streaming Multiprocessors (SM)



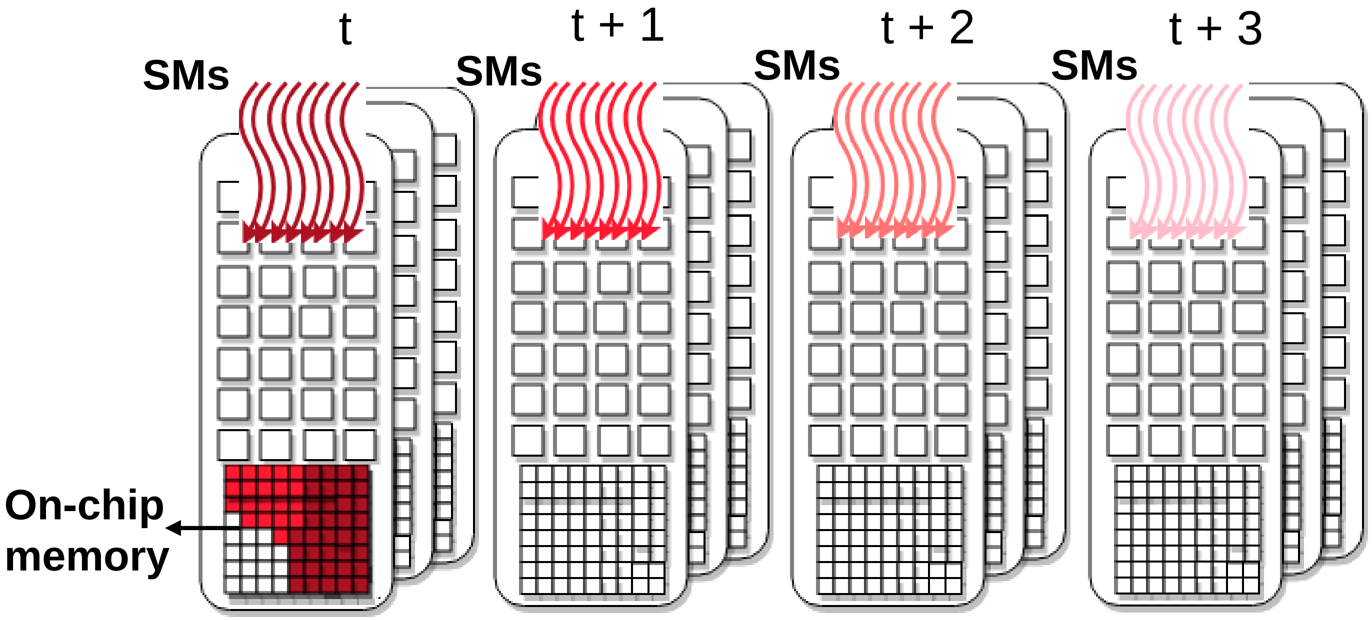
Proposal - Compute

- Send partially computed activation matrix from the producer SMs to the consumer SMs



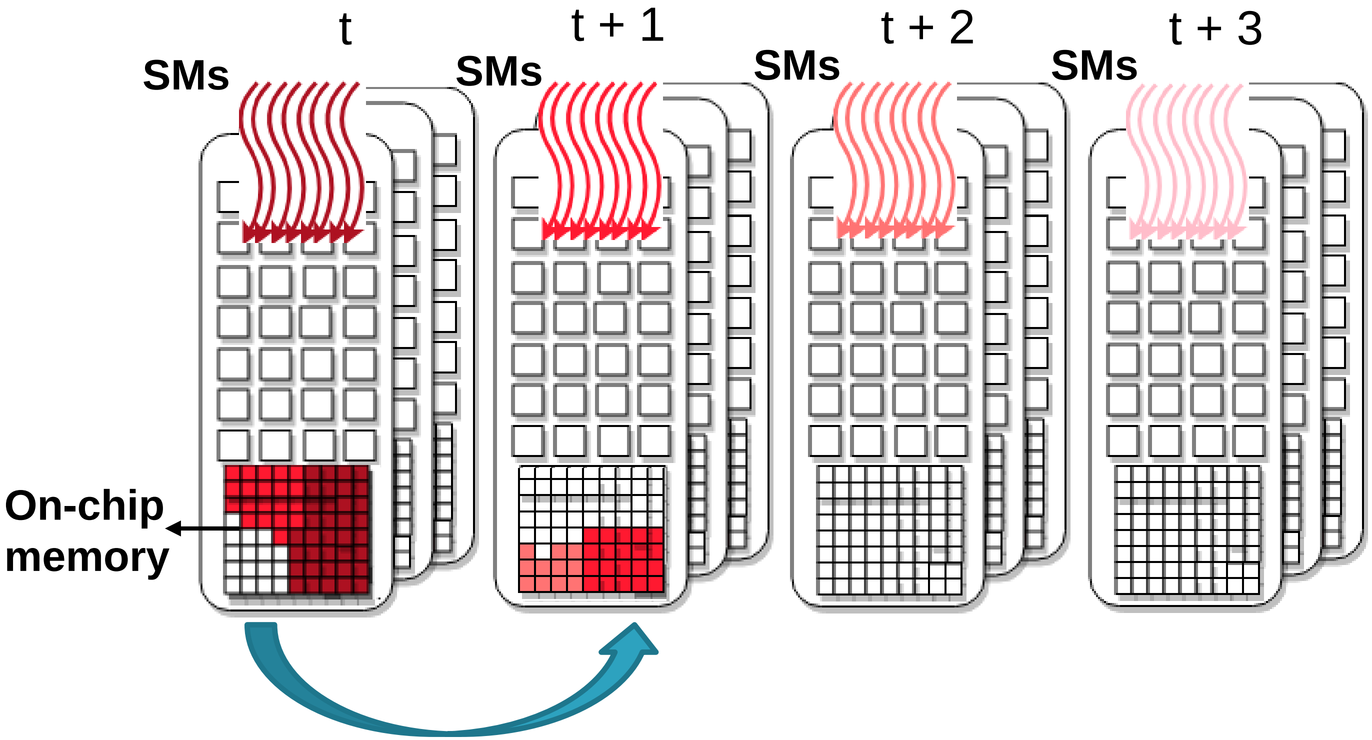
Proposal - Compute

- Send partially computed activation matrix from the producer SMs to the consumer SMs



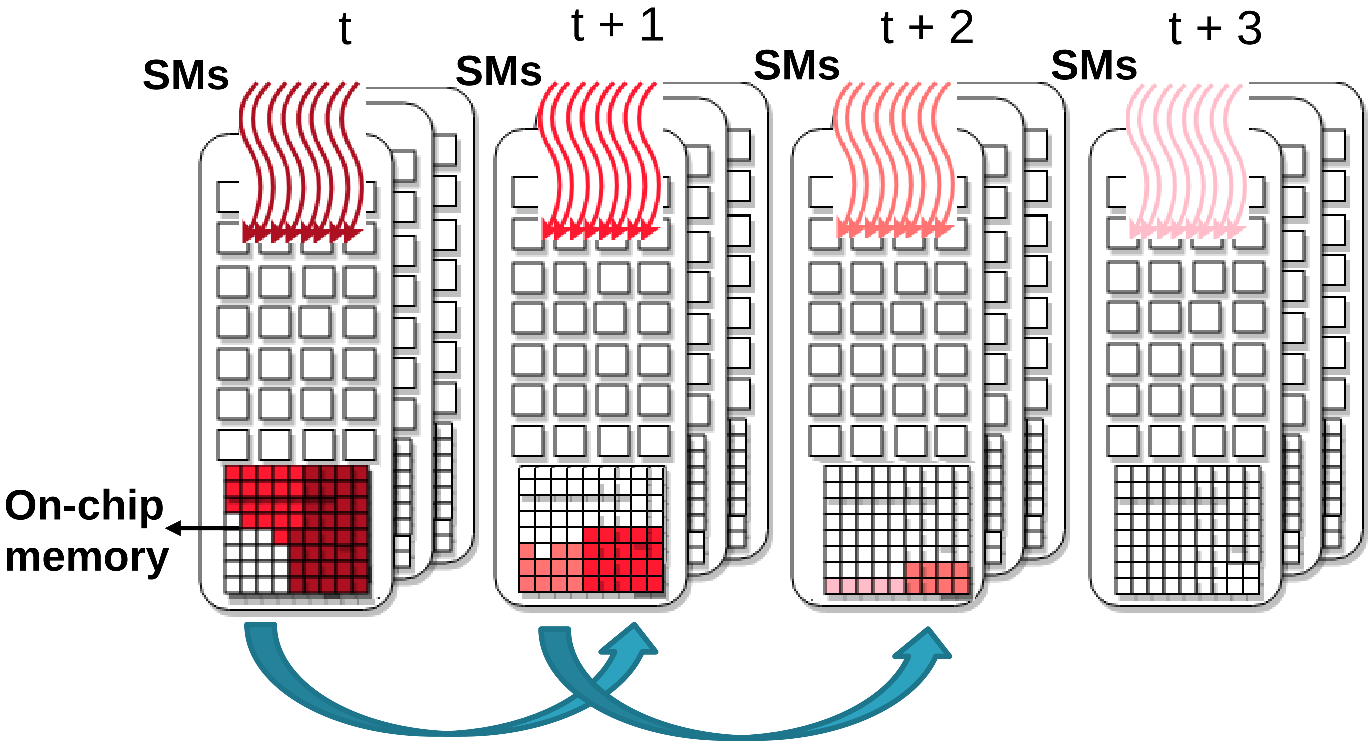
Proposal - Compute

- Send partially computed activation matrix from the producer SMs to the consumer SMs



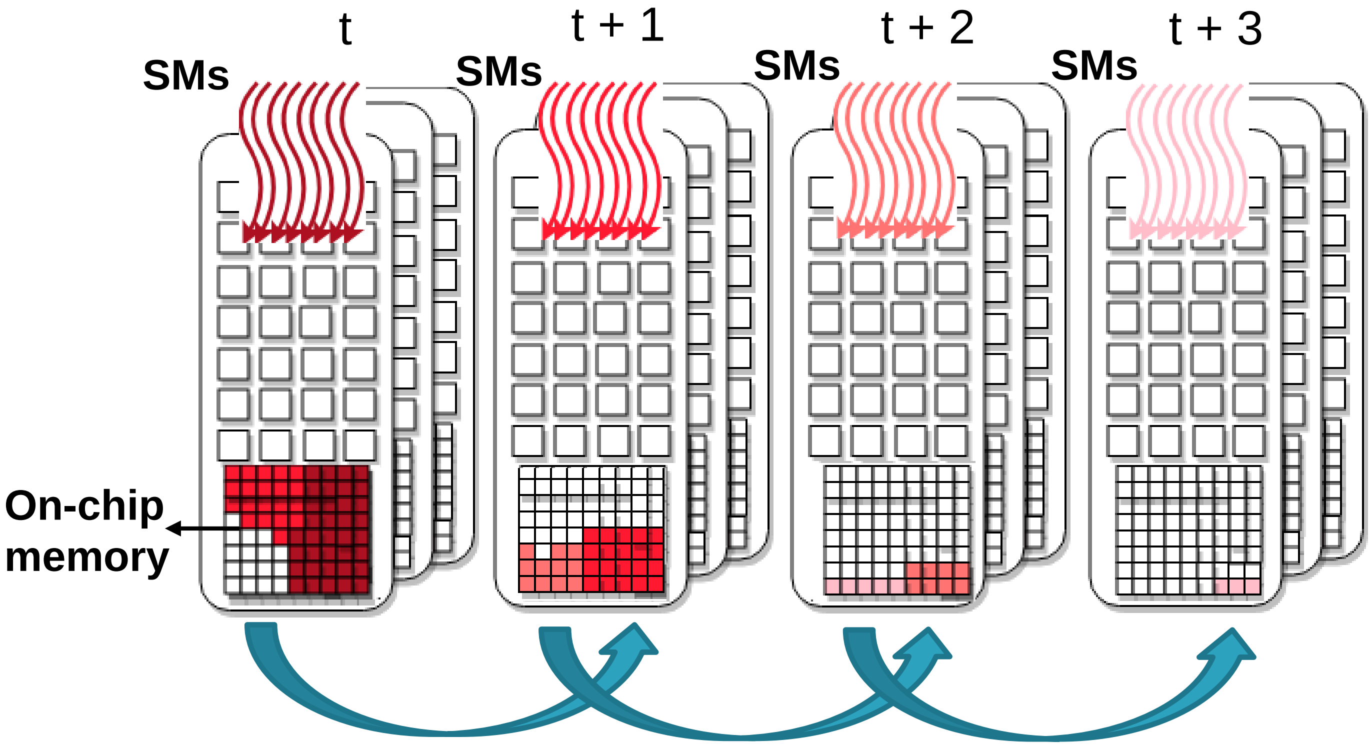
Proposal - Compute

- Send partially computed activation matrix from the producer SMs to the consumer SMs



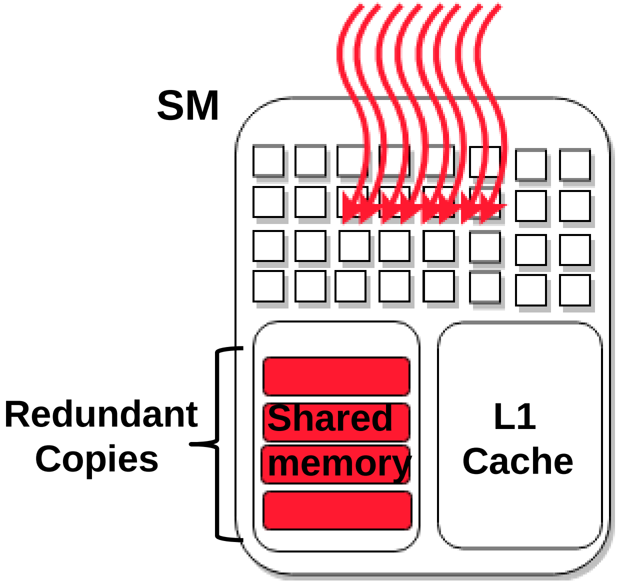
Proposal - Compute

- Send partially computed activation matrix from the producer SMs to the consumer SMs



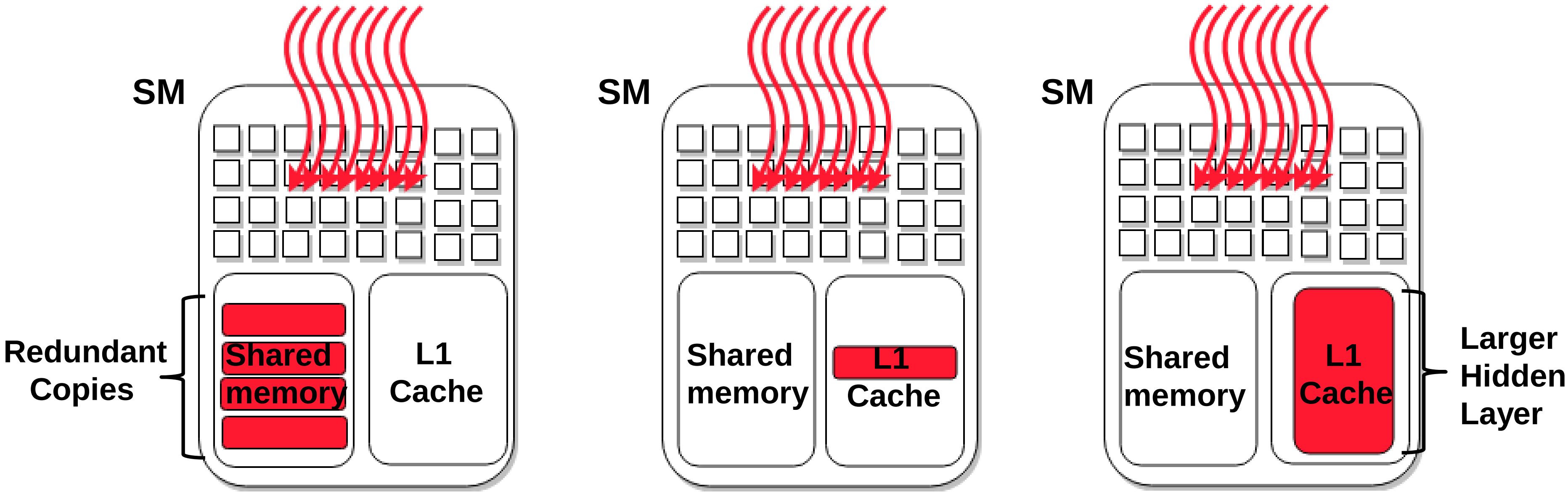
Proposal - Memory

- **Prior work:** Use shared memory for activations **[P-RNN ICML '16]**



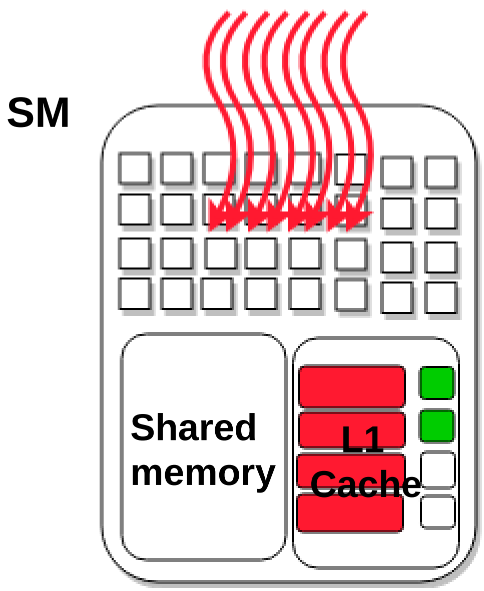
Proposal - Memory

- **Prior work:** Use shared memory for activations **[P-RNN ICML '16]**
- **We propose:** Use L1 cache for activations
  - **Enables larger recurrent layer** sizes
  - Requires only one copy



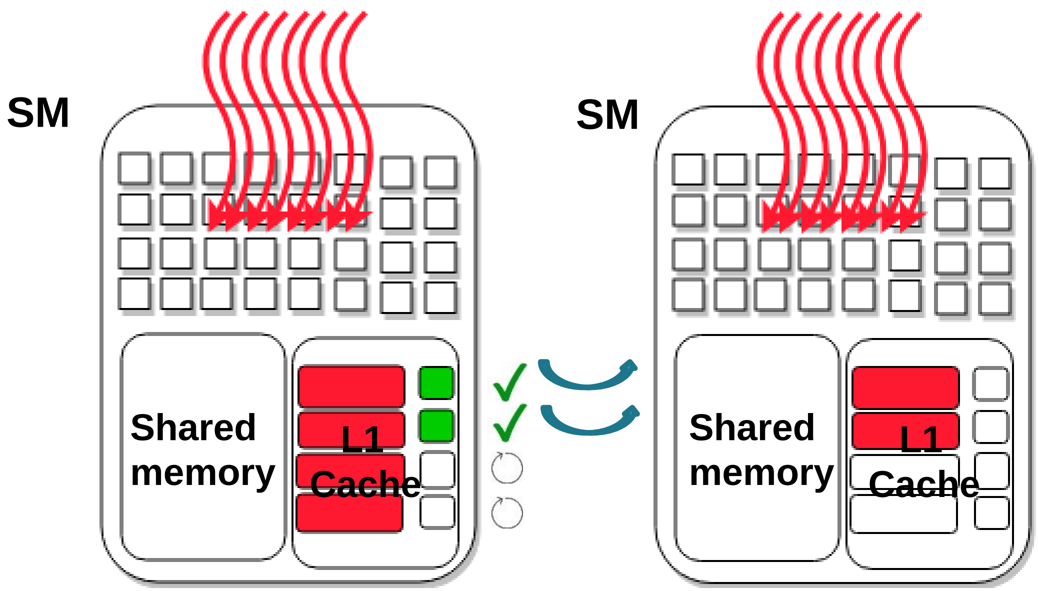
Proposal - Memory

- **We propose:** Use L1 cache for activations
  - **Reduce memory bandwidth**
    - locking cache ways
    - no need to synchronize with global memory



Proposal - Memory

- **We propose:** Use L1 cache for activations
  - **Reduce memory bandwidth**
    - locking cache ways
    - no need to synchronize with global memory
  - **Reduce Communication Overhead**
    - propagate updated activations directly to the consumers’ L1 cache



Use Stash [ISCA ’15]!

Evaluation

- **Simulation environment: GPGPU-Sim**
  - CUDA-based GPU simulator
  - Simulates Pascal and Volta architectures
  - Supports cuDNN and cuBLAS [ISPASS ‘19]
  - Ongoing work to enable execution of RNN kernels
- **Initial workload: DeepBench**
  - Training and inference of RNNs
  - Vanilla, LSTM and GRU
  - Varying hidden units, timesteps, batch size, seq. length.
- **Other workloads:**
  - DeepSpeech2, Persistent-RNN, Simple Recurrent Units (SRU)

Summary

- **RNN challenges:**
  - Dependencies between timesteps that limit parallelism
  - Poor temporal locality -- high memory bandwidth required

Summary

- **RNN challenges:**
  - Dependencies between timesteps that limit parallelism
  - Poor temporal locality -- high memory bandwidth required
- **Insight: exploit pipeline parallelism**
  - Extract parallelism through pipelining computations
  - Eliminate redundancy by using globally visible memory for activations
  - Optimize coherence protocol to exploit producer-consumer parallelism

Summary

- **RNN challenges:**
  - Dependencies between timesteps that limit parallelism
  - Poor temporal locality -- high memory bandwidth required
- **Insight: exploit pipeline parallelism**
  - Extract parallelism through pipelining computations
  - Eliminate redundancy by using globally visible memory for activations
  - Optimize coherence protocol to exploit producer-consumer parallelism
- **Potential to:**
  - Improve parallelism – increase GPU efficiency
  - Reduce memory bandwidth requirement – improve performance
  - Enable larger hidden layer RNN – enable better accuracy

Summary

- **RNN challenges:**
  - Dependencies between timesteps that limit parallelism
  - Poor temporal locality -- high memory bandwidth required
- **Insight: exploit pipeline parallelism**
  - Extract parallelism through pipelining computations
  - Eliminate redundancy by using globally visible memory for activations
  - Optimize coherence protocol to exploit producer-consumer parallelism
- **Potential to:**
  - Improve parallelism – increase GPU efficiency
  - Reduce memory bandwidth requirement – improve performance
  - Enable larger hidden layer RNN – enable better accuracy

Rethinking GPU architecture holistically can significantly improve performance and efficiency!!

QUESTIONS?