Tai Duc Nguyen - ECEC 623 - 03/08/2020

Academic Survey

This document includes the sumary of different academic papers which focus on Memory Controler (MC) optimizations

1. CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability
2. ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers
3. High-Performance and Energy-Efficient Memory Scheduler Design for Heterogeneous Systems
4. Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems
5. BLISS Balancing Performance, Fairness and Complexity in Memory Access Scheduling
6. EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM
7. A Memory Controller with Row Buffer Locality Awareness for Hybrid Memory Systems
8. Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability

---

- Research Team:
  - **ETH Zurich, CMU** - Hasan Hassan et. al.
- Type of Memory:
  - DRAM
- Research Focus:
  - CROW (CopyRow) DRAM is a new DRAM chip design + optimization in the MC that can improve the performance, energy eff, and reliability. This technology enables 2 implementations, which can be combined in 1 system, such as: CROW-cache and CROW-ref. CROW-cache implements a way to reduce latency of the most-recently-accessed rows by caching the data of those "regular rows" in "copy rows" (of the same bank), so that when data is accessed, both rows are activated at the same time, making charge-sharing more efficient, and hence, reduce latency. CROW-ref, on the other hand, uses efficient profiling to identify "weak rows" (rows that have low refresh interval), and mapped them to "strong copy rows" so that the refresh interval of the entire chip is much longer (2x-4x). These 2 CROW methods can be done with

minimal change: CROW-table in MC, 2 new commands (ACT-t & ACT-c) and 1 independent decoder (for the copyrows).

- Simulators:
  - SPICE (open sourced): github.com/CMU-SAFARI/CROW
  - CACTI (open sourced): www.hpl.hp.com/techreports/2009/HPL-2009-85.html
- Benchmarks:
  - SPEC CPU2006 www.spec.org/cpu2006/
  - TPC www.tpc.org/
  - STREAM github.com/jeffhammond/STREAM
  - MediaBench mathstat.slu.edu/~fritts/mediabench/
  - 2 synthetic applications: 1) random 2) streaming (not open sourced)

ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers

- Research Team:
  - **CMU** - Yoongu Kim et. al.
- Type of Memory:
  - DRAM
- Research Focus:
  - ATLAS (Adaptive per-Thread Least Attained-Service memory scheduling) is a new memory scheduling technique which improves system throughput in multi-processing-core-system without requring significant coordination among MCs. The technique takes both short-term and long-term thread behavior into account when ranking them. For short-term, it follows the LAS (least-attained service) rule, prioritizing threads that were least served in the last period. For long-term, it calculates the total attained service (from begining to time period i):
    - $total_{AS} = \alpha \times total_{AS-1} + \frac{(1-\alpha)}{thread\_weight} * AS_i$
  - Then the thread with the least $total_{AS}$ will be ranked highest.
  - Also, outstanding requests that have been waiting in the queue for more than T cycles will be prioritized over all others.
- Simulators:
  - In-house cycle-precise x86 CMP simulator (not open sourced)
- Benchmarks:
  - SPEC CPU2006 (410.bwaves, 416.gamess, 434.zeusmp are not included)

High-Performance and Energy-Efficient Memory Scheduler Design for Heterogeneous Systems

- Research Team:

    - **CMU, AMD Research, Intel Labs, Facebook, ETH Zurich** - Rachata Ausavarungnirun et. al
- Type of Memory:
    - DRAM
- Research Focus:
    - In heterogeneous systems where CPUs and GPU (on the same die), for example, share the same off-chip DRAM, the requests from the GPU can interfere with those from CPUs. Hence, the paper proposes SMS (Staged Memory Scheduler), which decouples the primary MC tasks into simpler structures. MC tasks are generally described as:
        - Detection of basic intra-application memory characteristics (row buffer locality)
        - Prioritization across applications & enforcement of policies
        - Low-level command scheduling, enforcement of DRAM timing constraints, and resolution of resource conflicts
    - Instead of all 3 tasks being performed at the same time, SMS divides 3 tasks into 3 stages:
        - **Batch Formation Stage**: Form batches for each CPU cores and GPU. Batches are ready when all requests (except first one) access the same DRAM row
        - **Batch Scheduling Stage**: This stage can be implemented with different existing scheduling policies (FRFCFS, PARBS, ATLAS), but the author propose one where SJF (Shortest Job First) policy is used with a probability p. If $p > 0.5$, then SJF is applied more often: applications with fewer outstanding requests are prioritized (CPU over GPU). If $p < 0.5$, then GPU is prioritized over CPU.
        - **DRAM Command Scheduler**: This stage is where the controller pop individual requests from batches received from the previous stage and schedule them according to timing constraints.
- Simulators:
    - In-house cycle-accurate simulator (not open sourced)
- Benchmarks:
    - 16 benchmarks from SPEC CPU2006 + 1 GPU application.

Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems

- Research Team:
    - **Microsoft Research** - Onur Mutlu
- Type of Memory:
    - DRAM
- Research Focus:

    - The authors propose PAR-BS (Parallelism-Aware Batch Scheduling), a new memory scheduler which provides quality-of-service (Qos), system throughput, and fairness in highly parallelized computer systems. In PAR-BS, there are 2 stages: Batch Formation and Request Prioritization. The MC forms new batch when there are no "marked" requests left in the memory request buffer (MRB). This new batch consists of outstanding requests in the MRB that were not included in the last batch. The MC will "mark" up to `MarkingCap` number of requests in the current batch. Then, it will prioritize requests in the following order:
        - Marked requests
        - Row hit
        - Higher rank (enable operator/programmer to have control)
        - Older request
    - The rank of a request is computed such that the shortest-job rule is obeyed: The thread with lower number of marked requests (in current batch) are ranked first. If there is a tie breaker, then the thread with the lower total number of marked request (over multiple batches) is prioritized.
- Simulators:
    - Cycle-accurate x86 CMP simulator based on Pin and iDNA
- Benchmarks:
    - SPEC CPU2006
    - Matlab
    - XML Parsing

BLISS Balancing Performance, Fairness and Complexity in Memory Access Scheduling

- Research Team:
    - **CMU** - Lavanya Subramanian et. al.
- Type of Memory:
    - DRAM
- Research Focus:
    - The author built a new application scheduling policy, BLISS (Blacklisting Memory Scheduler), which achieves higher system performance, better fairness, and less hardware complexity in a highly parallelized computer system. The scheduler works by implement a new counter variable with each application. If the application be served currently is the same as the application being served immediately before, then the counter for such application is increment by 1. If this counter is greater than the `BlacklistingThreshold`, then the application is deemed `blacklisted`. Hence, the prioritization of memory requests are as follow:
        - Non-blacklisted application
        - Row buffer hit
        - Older request
    - The blacklisting information is reset every X cycles.

- Simulators:
    - In-house cycle-precise simulator similar to Ramulator
    - Synopsis Design Compiler (for hardware cost estimation) with 32nm standard cell library
- Benchmarks:
    - SPEC CPU2006
    - TPC-C
    - Matlab
    - NAS parallel benchmark suite

EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM

- Research Team:
    - **ETH Zurich** - Skanda Koppula et. al.
- Type of Memory:
    - DRAM
- Research Focus:
    - The authors propose a method to run an **already trained** DNN using conventional DRAM (or any kind of memory which can trade power for Bit Error Rate) that can consume less power and faster. This method comprises of 3 stages: Error Boosting, DNN characterization, and DNN to DRAM mapping. Error Boosting is a step where small incremental errors (bit flips) are introduced into the DNN parameters until the DNN is deemed `unreliable`. This step is performed in concurrent with correcting implausible values (zero-out values that are Out-Of-Bounds) to avoid *accuracy collapse*. After the network finishes Error Boosting, the highest BER of the entire DNN (Coarse-grained Characterization), or of each DNN datatypes (int4, int8, etc.) (Fine-grained Characterization) is determined. These BERs are then used to map the entire DNN (Coarse-grained Mapping), or parts of the DNN (Fine-grained Mapping) to regions in the DRAM which satisfy the BER requirements. This process is performed with a priori knowledge on the DRAM's timing characteristics, which are acquired by performing a series of tests where the voltage is reduced in a region and a numeric pattern is read out. In order to *Correct Implausible Values*, and enable *Coarse/Fine-grained Mapping*, the author introduces complexity into the Memory Controller so that it can correct values and change voltage and timing parameters in an online manner.
- Simulators:
    - FPGA running SoftMC
    - PyTorch
- Benchmarks:
    - CIFAR-10

    - ILSVRC2012
    - ResNet101
    - VGG-16
    - DenseNet201
    - Google MoblileNetV2
    - SqueezeNet
    - YOLO/YOLO-Tiny

A Memory Controller with Row Buffer Locality Awareness for Hybrid Memory Systems

- Research Team:
    - **Google, CMU, Facebook, MIT, ETH Zurich** - HanBin Yoon et. al.
- Type of Memory:
    - Hybrid DRAM + NVM (PCM)
- Research Focus:
    - The authors expand on the idea where DRAM is used as a "fast" cache in hybrid Memory System by introducing a Dynamic Row Buffer Locality Awareness (DRBLA) system which can adapt the `MissThreshold` in an online manner. This threshold is used to determine if a row should be remapped from NVM to DRAM so that further row buffer miss will result in less latency. The Memory Controller will find the best `MissThreshold` by performing a simple hill-climbing algorithm to find the sweet spot where the benefit of caching the data is greater than the cost of migrating the data.
- Simulators:
    - In house x86 multicore simulator (predecessor of Ramulator and ThyNVM simulator)
- Benchmarks:
    - Cloud/Server type applications: TPC-C/H
    - Apache Web ServerV
    - Video processing benchmarks

Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

- Research Team:
    - **Cornell University, Microsoft Research** - Engin Ipek et. al.
- Type of Memory:
    - DRAM
- Research Focus:
    - The authors introduce a new DRAM Memory Controller which only uses Reinforcement Learning (RL) to intelligently schedule memory commands. With any RL system, 3 considerations must be analyzed: 1) Credit Assignment Problem, 2)

Balancing Exploration & Exploitation, and 3) Generalization. Since the memory scheduling is an ***infinite-horizon*** process, it need necessary for the cumulative reward function to have a discount parameter (γ) so that convergence is mathematically possible.
        - $E[\sum_{i=0}^{\inf} \gamma^t \cdot r_{t+i}]$ where E is the cumulative reward function, $0 \leq \gamma \leq 1$, and r is the immediate reward value
    - In order to choose an action that leads to the reward r, however, the expected value of the cumulative reward, Q, must be computed. The author uses the learning algorithm SARSA, which update the Q for every state-action pairs by:
        - $Q(s_{prev}, v_{prev}) = (1 - \alpha)Q(s_{prev}, v_{prev}) + \alpha[r + \gamma Q(s_{current}, v_{current})]$. The MC will chose to the perform the legal action whose Q value is largest given its current state.
    - To enable exploration, however, the MC will take a random action at a probability ϵ.
    - Lastly, in order for the RL algorithm to generalize and the system to conserve computation/memory usage, the large number of states are quantized (clustered) in to cells, which form tables. Then, CMAC (Cerebellar Model Arithmetic Computer) learning model will shifts these "coarse-grained" tables by a random amount with respect to each other so that some states in different tables will share the same Q value. This method, in conjunction with an efficient hashing algorithm makes the computation very fast and conserves memory usage.
- Simulators:
    - In-house x86 simulator (modified version of SESC with Intel Nehalem arch)
- Benchmarks:
    - SCALPARC
    - MG, CG
    - SWIM-OMP
    - EQUAKE-OMP
    - ART-OMP
    - OCEAN
    - FFT
    - RADIX