Background and Problems

- DRAM cells require **periodic refresh** to prevent data loss from leakage
- **Problems:**
  1. System performance degradation
     - **All-bank refresh (REFab):** memory controllers refresh every bank within a rank, blocking the rank from servicing memory requests
  2. DRAM scaling
     - As DRAM density increases (more cells), refresh latency is expected to increase

**Dynamic Access Refresh Parallelization (DARP):**

1. **Refresh scheduling policy with two components**
   1. **Out-of-order per-bank refresh (REFpb):** refresh one bank at a time, following a strict sequential round-robin order
   2. **Write-refresh parallelization:**
      - **Key observations:**
        1) Write requests are buffered and drained to DRAM in a batch
        2) Write requests are not latency-critical
      - **Key idea:** select the bank with the fewest number (or none) of pending writes to refresh while DRAM is draining writes

2. **Subarray Access Refresh Parallelization (SARP):**

   - **Key observations:**
     1) A bank consists of multiple subarrays (sub-banks)
     2) Every subarray has its own local sense amplifiers (row buffer) to perform refresh operations
     3) DRAM I/O remains idle under refresh
   - **Key idea:** enable a bank to service accesses in idle subarrays in parallel with refreshes to other subarrays in a bank

Our Solution

- **Our goal:** improve system performance over existing refresh schemes by mitigating refresh penalty
- **Our key idea:** hide refresh latency by parallelizing refresh operations with memory accesses to avoid delaying memory requests

Results

**Methodology**

- 8 OoO cores, 4GHz, 3-wide issue
- 64KB L1, 512KB private L2 cache slide/core
- Memory controller: 64-entry request queue, FR-FCFS scheduling
- DRAM: DDR3-1333, 2 channels, 2 ranks/channel, 8 banks/rank, 8 subarrays/bank
- Simulation: cycle-level x86 multi-core simulator
- Workloads: TPC, STREAM, SPEC CPU2006

**Dynamic Access Refresh Parallelization (DARP):**

1. **Out-of-order per-bank refresh (REFpb):**
   - **Key observation:** DRAM has internal logic that strictly refreshes banks in a round-robin order
   - **Key idea:** refresh banks in out-of-order fashion by issuing a per-bank refresh to any idle bank

2. **Write-refresh parallelization:**
   - **Key observations:**
     1) Write requests are buffered and drained to DRAM in a batch
     2) Write requests are not latency-critical
   - **Key idea:** select the bank with the fewest number (or none) of pending writes to refresh while DRAM is draining writes

**Subarray Access Refresh Parallelization (SARP):**

- **Key observations:**
  1) A bank consists of multiple subarrays (sub-banks)
  2) Every subarray has its own local sense amplifiers (row buffer) to perform refresh operations
- **Key idea:** enable a bank to service accesses in idle subarrays in parallel with refreshes to other subarrays in a bank
- **Area overhead:** 0.71% based on Rambus DRAM model