A High-Performance Software Framework for Memory on Commodity GPUs

Naoya Maruyama, Akira Nukada, Satoshi Matsuoka (Tokyo Institute of Technology)

Contact: naoya@matsulab.is.titech.ac.jp

Background

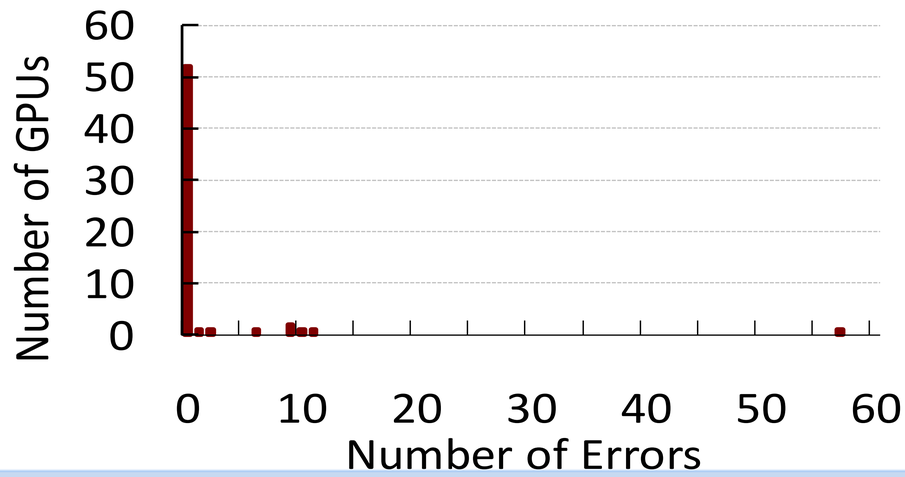
Problem: No ECC for GPU DRAM (yet?)

- Radiation-induced bit-flip errors are inevitable
  - SEC-DED ECC in DRAM is a “must” in server memory
- Redundancy must be implemented

Observation & Idea

- Compute-intensive kernels → Little performance effects
- Memory-intensive kernels → Compute units are available
  - *Exploit idle units for data coding*

Histogram of the number of bit-flips on GeForce 8800 GTS 512 running Memtest for 72 hours



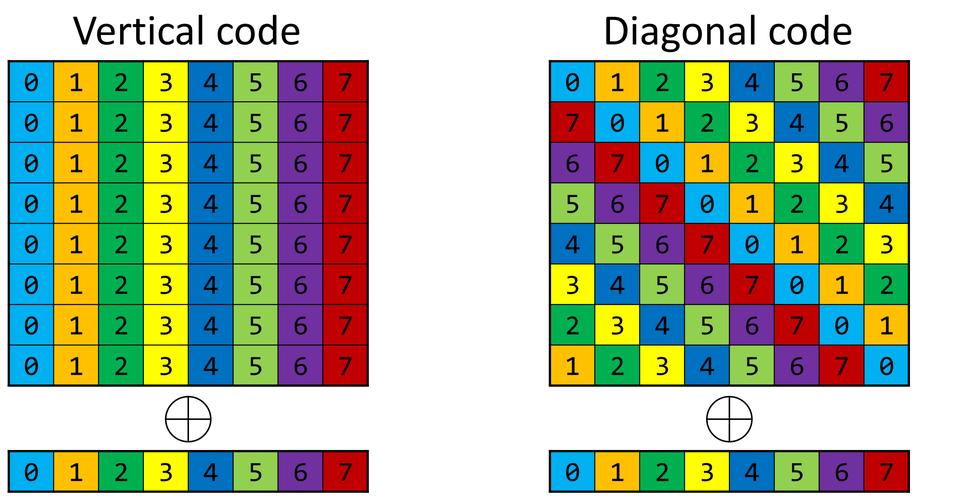
Method

Software-only framework for DRAM error tolerance

- Error detection in CUDA global memory + Checkpoint/Restart
- Works with existing NVIDIA CUDA GPUs

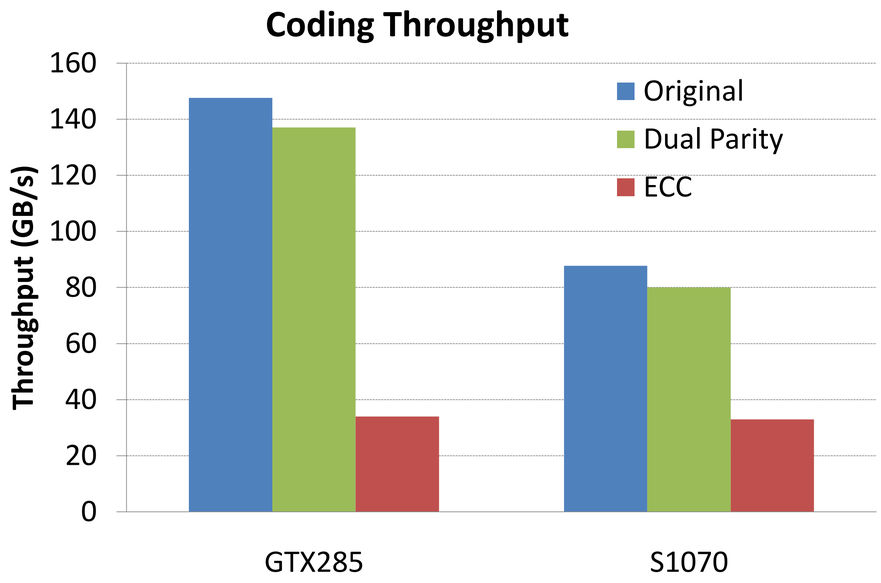
Lightweight Error Detection

- “Dual Parity” for 128B blocks of data
  - Detects a single-bit error in a 4B word
  - Detects a two-bit error in a 128B block
  - No on-the-fly correction → Rollback upon errors
- Per-block xor reduction exploiting shared memory
- Generates a *vertical* parity and *diagonal* parity for each block
  - Vertical parity: 4B of bits to store xor of 32 4B words
  - Diagonal parity: 4B of bits to store xor of 32 rotated 4Bwords



Integration to Applications

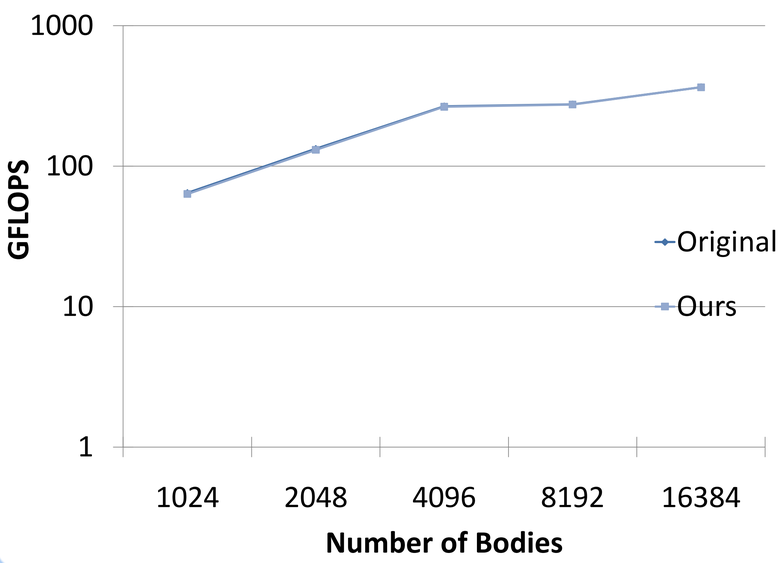
- Input transfer
  - Generates dual-parity codes on host and transfer them to global memory
  - Operates in the background of the original data transfer to hide the overhead
- Kernel execution
  - Global memory reads: detect errors by computing the dual-parity codes for the read data
  - Global memory writes: write dual parity for the written data
- Output transfer
  - Detects errors by running a separate error-checking kernel after the original kernel execution



Results

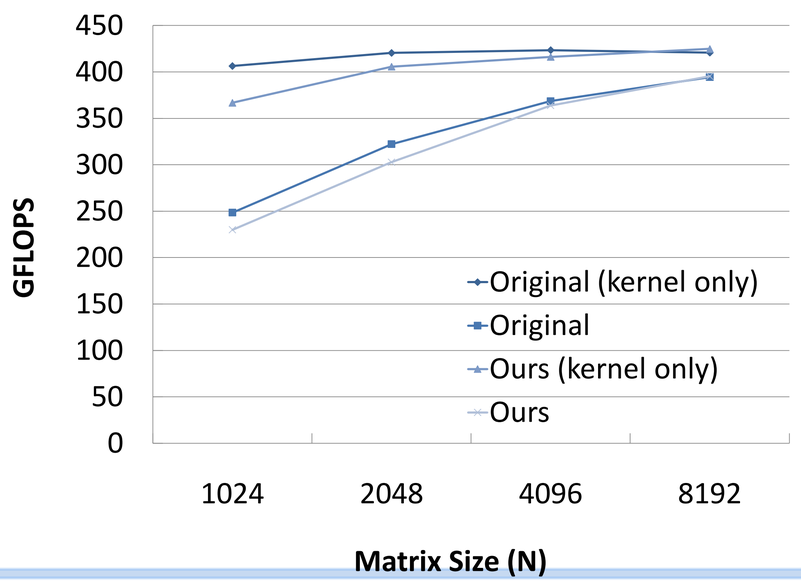
N-Body Problem

- Checks current positions and velocities by a separate kernel
- Encodes new positions and velocities
- Highly compute intensive kernel
- *2% overheads at most*



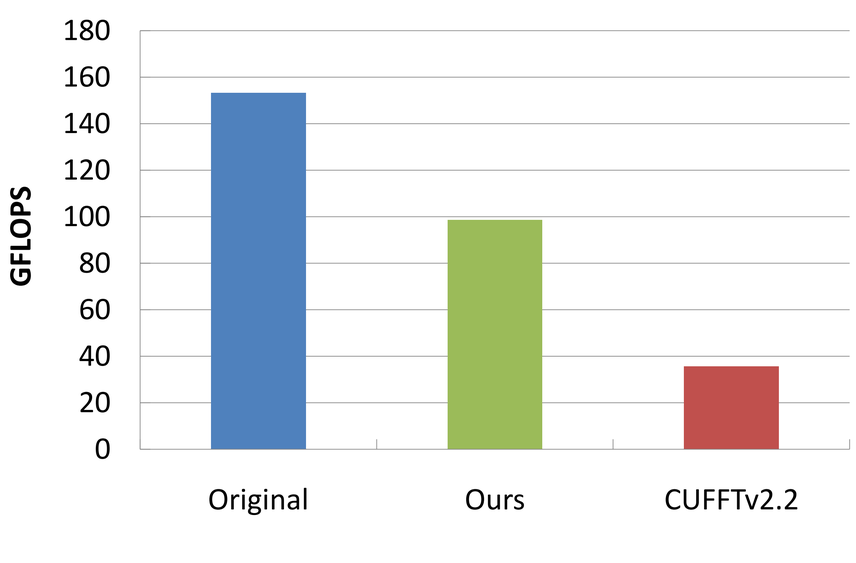
Matrix Multiplication

- Checks two input matrices by a separate kernel
- Encodes results
- Compute intensive kernel
- *<10% overheads*



256^3 3D FFT

- Inline checking of read data
- Encodes transformed data
- Bandwidth intensive but compute resources are also exercised
- *35% overheads*



NVIDIA Research Summit’09, Sep 2009.