# Enabling Dynamic Memory Management Support for MTC on NVIDIA GPUs


**Benjamin Grimmer, Scott Krieder, Ioan Raicu**
Dept. of Computer Science
Illinois Institute of Technology
{bgrimmer, skrieder}@hawk.iit.edu, iraicu@cs.iit.edu


## Overview

- MTC workloads are poorly supported on current NVIDIA GPUs.
- Aim to meet the dynamic memory management needs of MTC applications on GPUs.
- Achieve this by allocating GPU memory through CUDA then sub-allocate it to tasks as needed.

## Many-Task Computing

- Bridges the gap between High Performance Computing (HPC) and High Throughput Computing (HTC)
- Many resources over short time
- Many computational tasks
- Tasks both dependent or independent
- Tasks workloads are organized as Directed Acyclic Graphs (DAG)
- Primary Metrics are measured in seconds

## Sub-Allocator Design

### Data Structures Usage:

- Maintain a circular linked list of free memory on GPU
- List is ordered by device address of memory chunk

### Allocating Memory:

- Uses First Fit.
- cudaMalloc() if no chunks are large enough.
- Writes a header preceding result

### Freeing Memory:

- Read header information
- Find correct list location
- Add to list and coalesce this memory with adjacent chunks

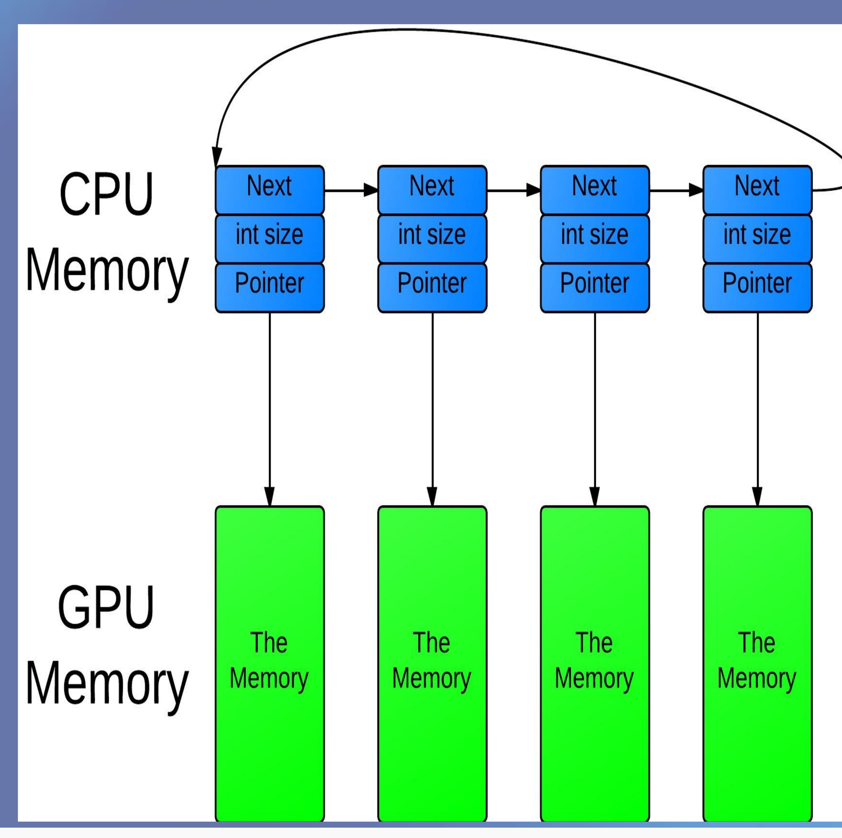


Fig 1. –Linked list of pointers to chunks of free GPU memory waiting to be sub-allocated

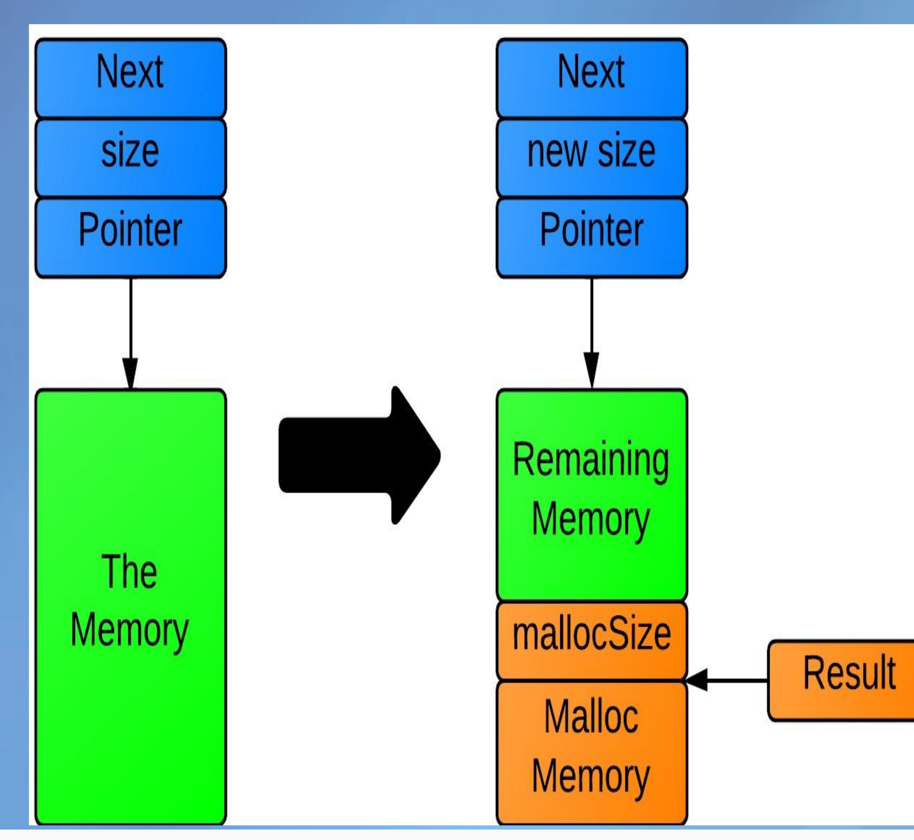


Fig. 2 – malloc operation, reducing the size of a chunk and writing a header into GPU memory.

## Proposed Work

This work aims to enable efficient dynamic memory management on NVIDIA GPUs by utilizing a sub-allocator between CUDA and the programmer. This work enables Many-Task Computing applications, which need to dynamically allocate parameters for each task, to run efficiently on GPUs.

## Sub-Allocator Preliminary Results

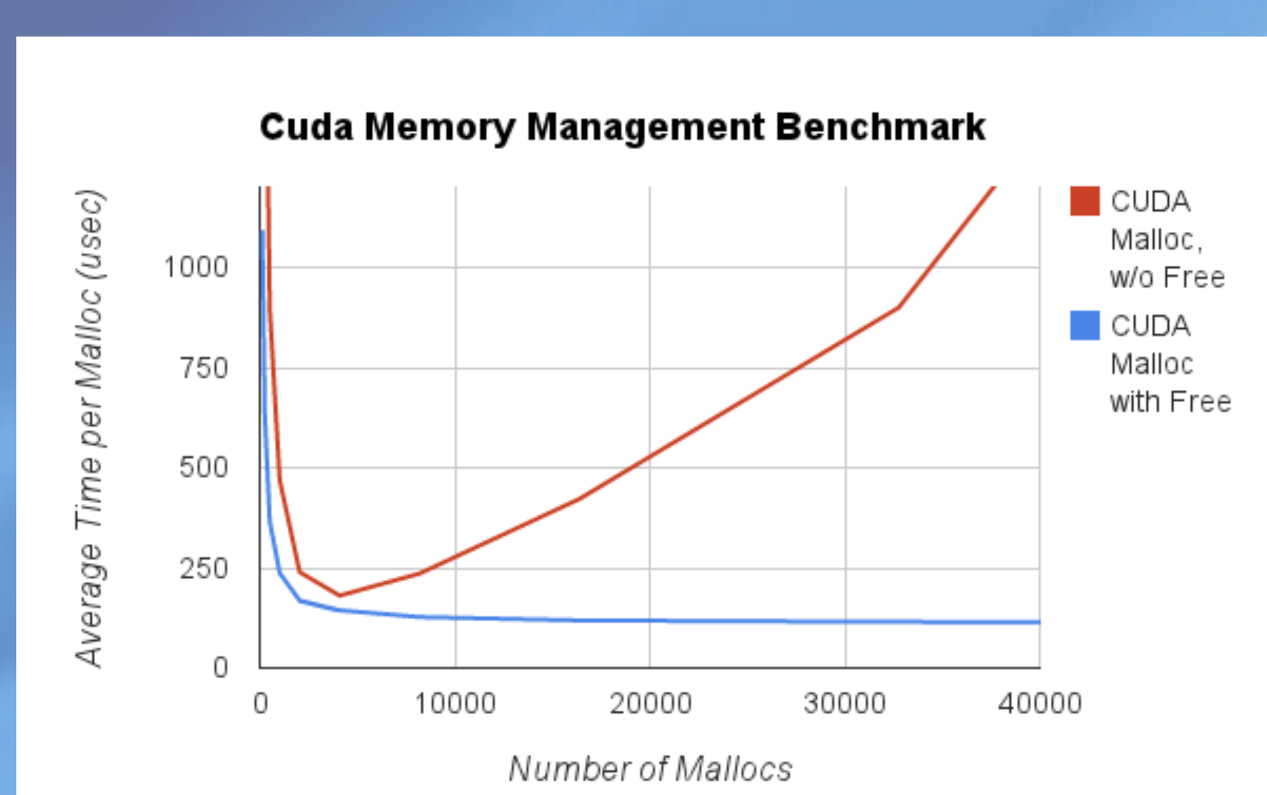


Fig 3. –CUDA memory management execution times, non-constant scaling after many mallocs

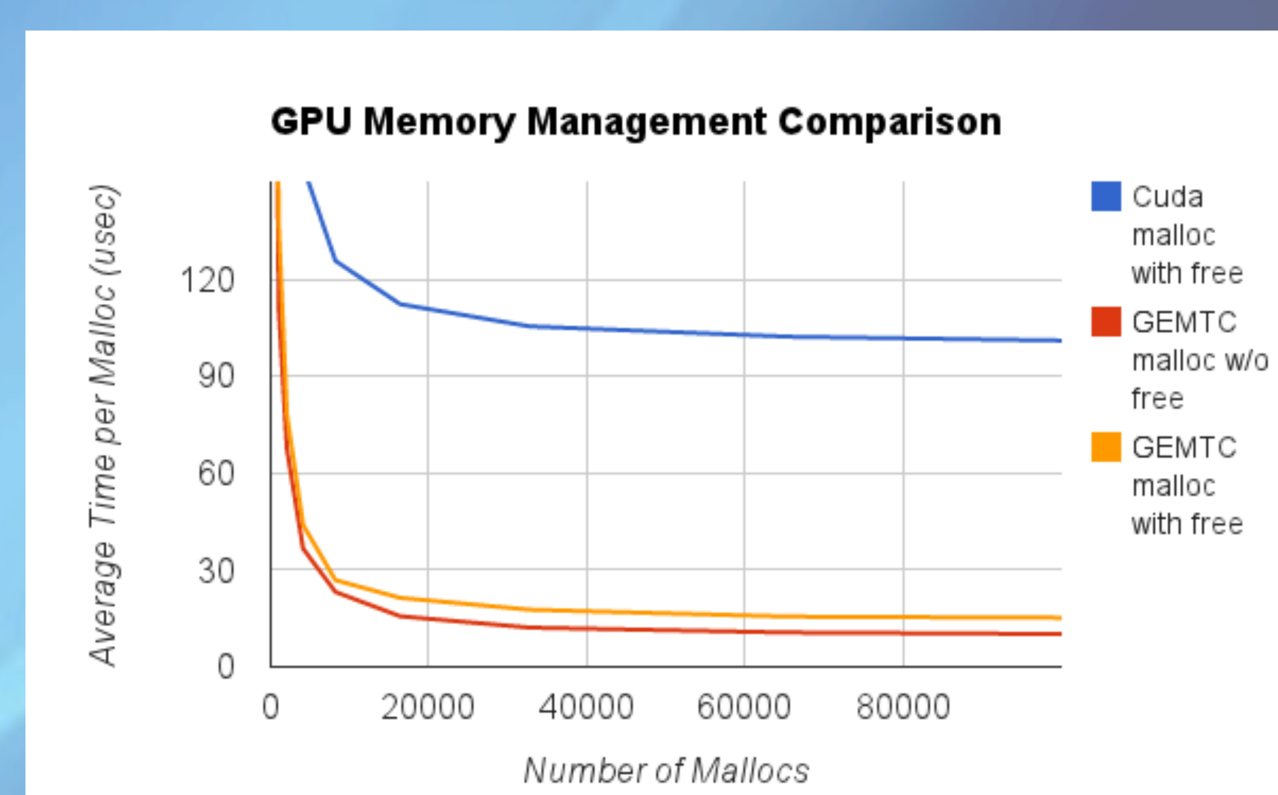


Fig. 4 – Sub-Allocator can do memory allocations and frees in ~14usec (10x faster than CUDA)

## Conclusions

- Improves CUDA's memory management for highly dynamic workloads.
- Constant scaling of allocation times after many calls to malloc.
- 8x speedup over CUDA in workloads with large number of malloc/free pairs.
- 30x speedup over CUDA after 10,000 calls to malloc.
- 100x speedup over CUDA after 30,000 calls to malloc.

## Future Work

- Improve the worst case time complexity of malloc and free operations. Currently both are O(n), where n is the number of memory fragments in our list.
- Change data structure from linked list to something providing O(log n) insertion and deletion.
- Evaluate and optimize the sub-allocator on K20 GPUs (currently tested on GTX670)
- Evaluate and optimize the sub-allocator for CUDA 5.0 (currently tested on CUDA 4.2)

## References


**GeMTC** – http://datasys.cs.iit.edu/projects/GeMTC
**NVIDIA** - nvidia.com/object/cuda_home_new.html