Towards the Support for Many-Task Computing on Many-Core Computing Platforms

Scott J. Krieder
Dept. of Computer Science
Illinois Institute of Technology
skrieder@iit.edu

Dr. Ioan Raicu
Dept. of Computer Science
Illinois Institute of Technology
iraicu@cs.iit.edu

DataSys Data-Intensive Distributed Systems Laboratory

ILLINOIS INSTITUTE OF TECHNOLOGY

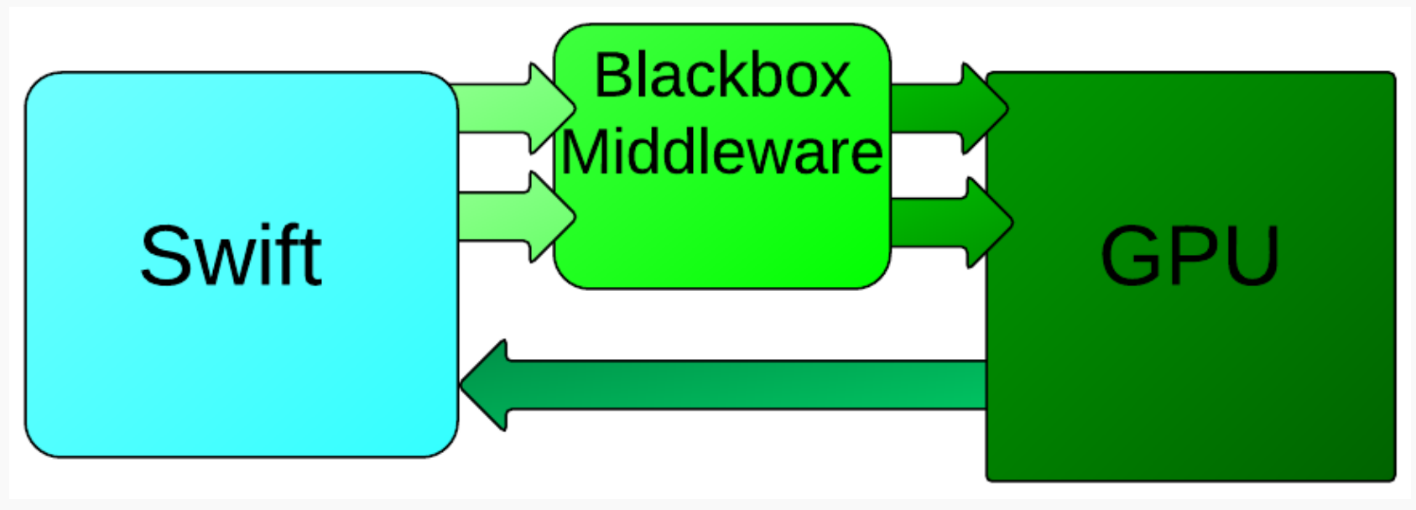
Overview

- MTC workloads are poorly supported on current accelerator platforms
- Aim to provide MTC support, easier programmability, higher efficiency.
- GeMTC framework provides many independent workers, efficient MTC workload support, and more efficient dynamic memory management.

Scheduler Designs and Motivation

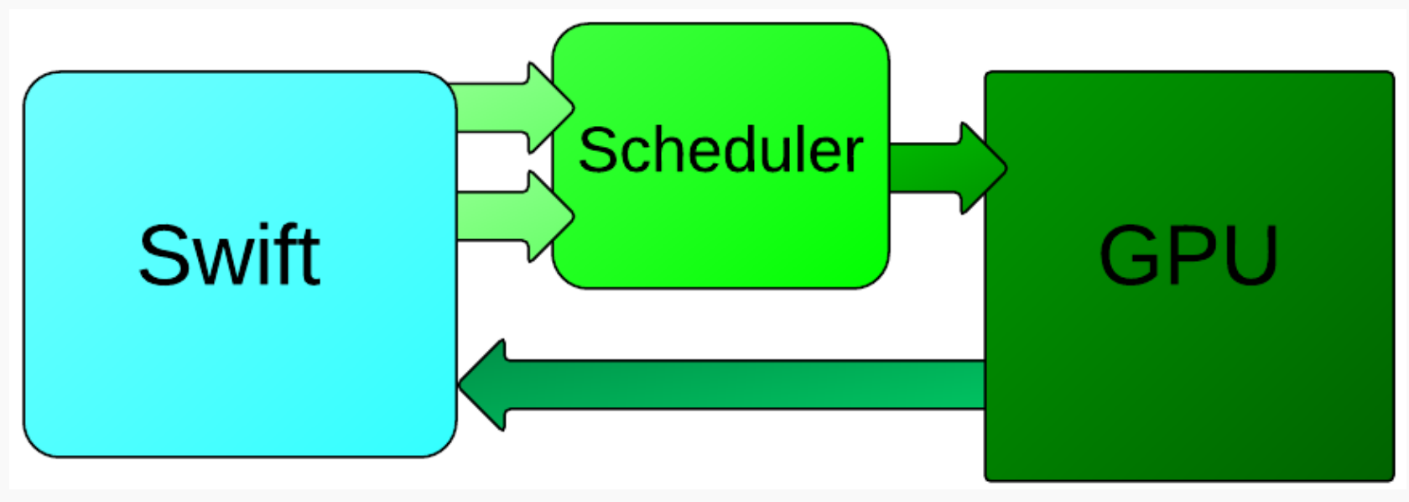
Naïve

- Sequential
- Time Shared



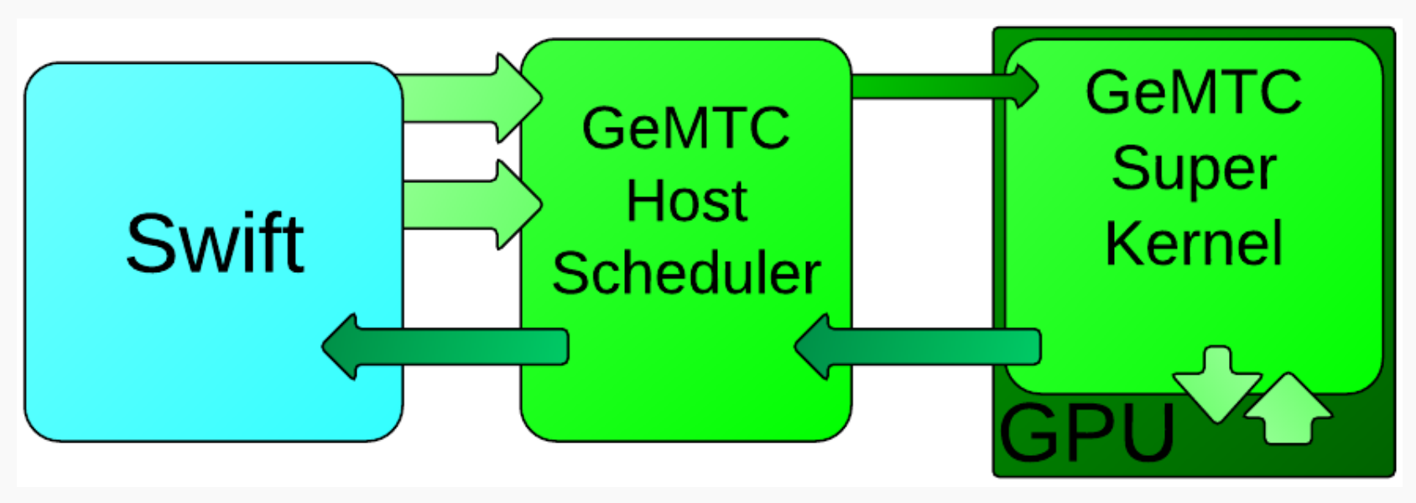
Batch

- Better
- Homogenous Tasks



GeMTC

- Heterogeneous Tasks
- Many Ind. Workers



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

GeMTC Preliminary Results

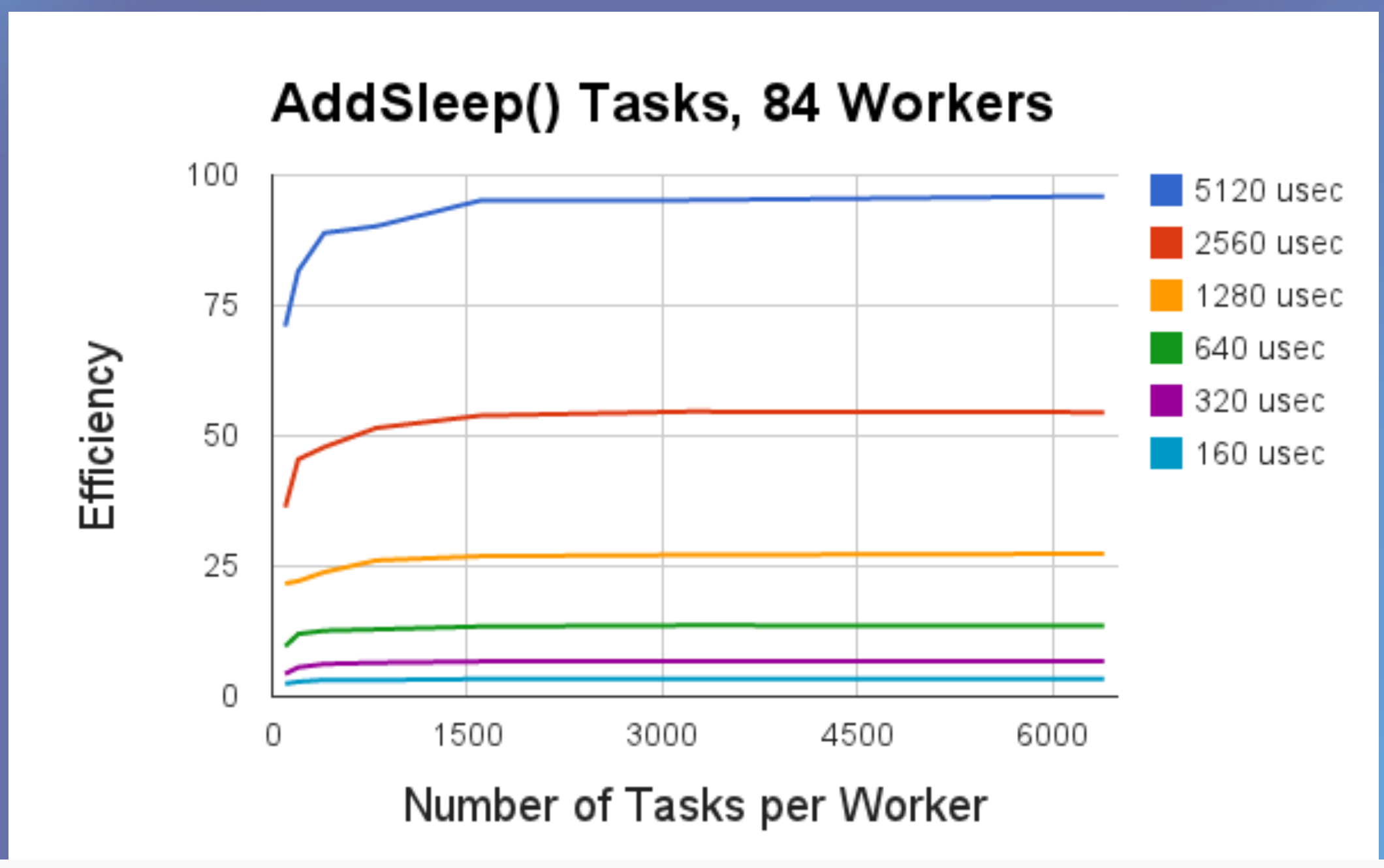


Fig 1. - Tasks >5 milliseconds see extremely high efficiency (~60 usec *80 workers = 4,800 ~= 5ms)

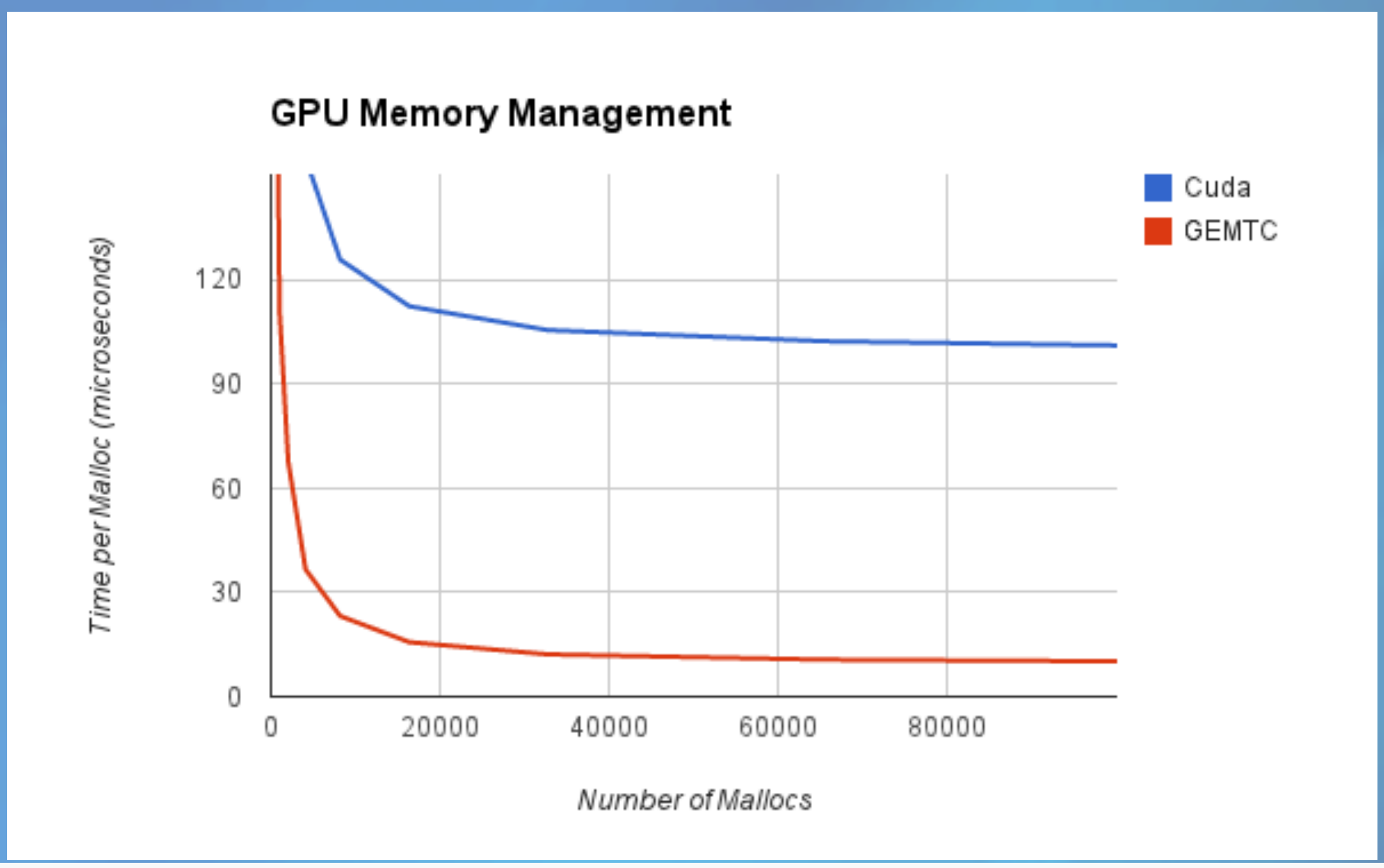


Fig. 2 - gemtcMalloc() outperforms cudaMalloc() Providing allocation in ~14 usec

Proposed Work

This work aims to address the programmability gap between MTC and accelerators, through an innovative middleware that enables MIMD workloads to run on SIMD architectures. This work will enable a broader class of applications to leverage the growing number of accelerated high-end computing systems.

Swift/T & Applications

- Swift/T is a parallel programming framework
- Scales up to X number of nodes with X number of cores
- Applications from many different domains already work with Swift and we are working to support them on GPUs running the GeMTC Framework.
- Application Domains include: Astronomy, Biochemistry, Bioinformatics, Climate Modeling, Economics, Finance, Medical Imaging.

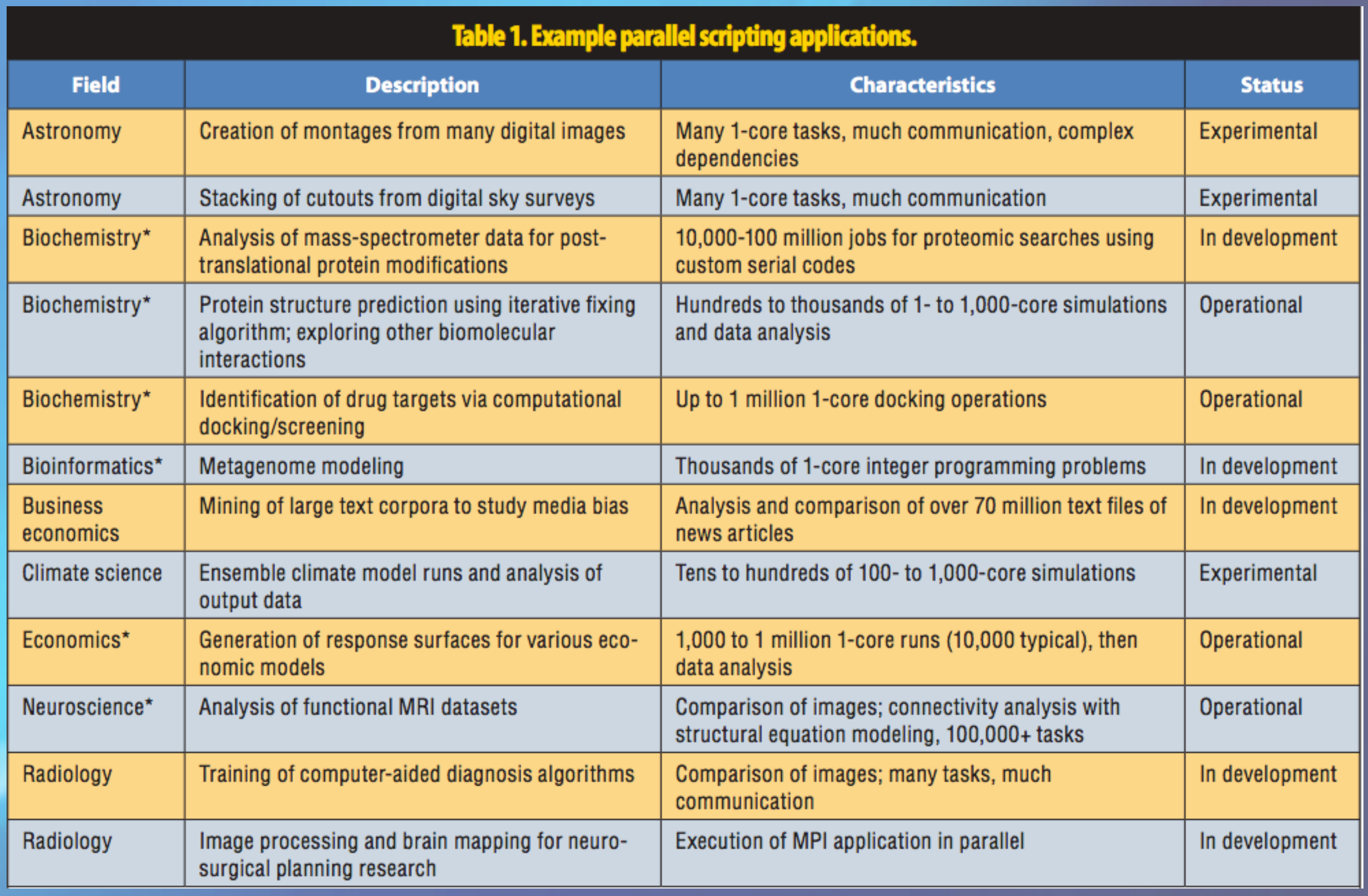


Table 1. Example parallel scripting applications.

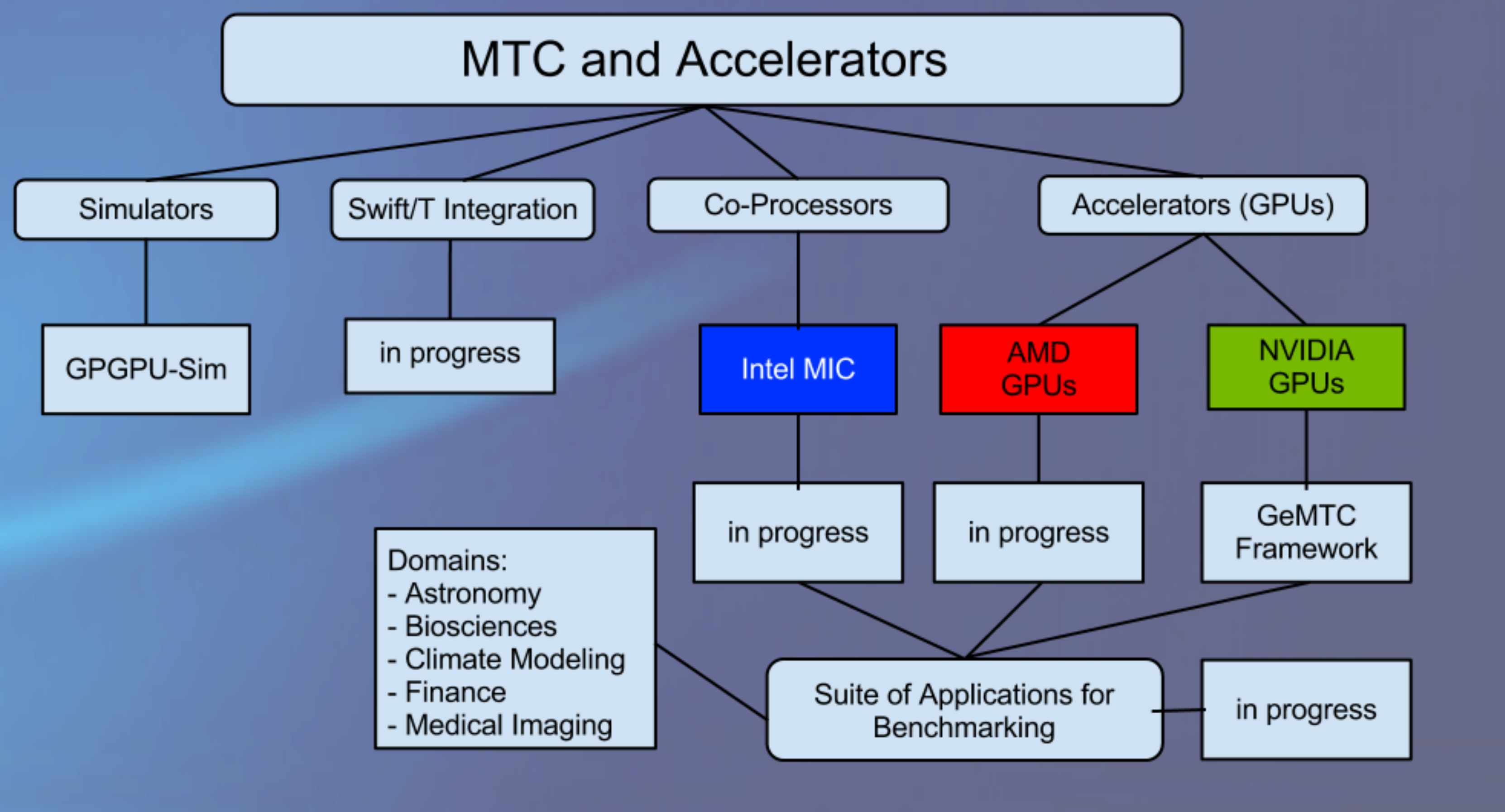
Field	Description	Characteristics	Status
Astronomy	Creation of montages from many digital images	Many 1-core tasks, much communication, complex dependencies	Experimental
Astronomy	Stacking of cutouts from digital sky surveys	Many 1-core tasks, much communication	Experimental
Biochemistry*	Analysis of mass-spectrometer data for post-translational protein modifications	10,000-100 million jobs for proteomic searches using custom serial codes	In development
Biochemistry*	Protein structure prediction using iterative fixing algorithm; exploring other biomolecular interactions	Hundreds to thousands of 1- to 1,000-core simulations and data analysis	Operational
Biochemistry*	Identification of drug targets via computational docking/screening	Up to 1 million 1-core docking operations	Operational
Bioinformatics*	Metagenome modeling	Thousands of 1-core integer programming problems	In development
Business economics	Mining of large text corpora to study media bias	Analysis and comparison of over 70 million text files of news articles	In development
Climate science	Ensemble climate model runs and analysis of output data	Tens to hundreds of 100- to 1,000-core simulations	Experimental
Economics*	Generation of response surfaces for various economic models	1,000 to 1 million 1-core runs (10,000 typical), then data analysis	Operational
Neuroscience*	Analysis of functional MRI datasets	Comparison of images; connectivity analysis with structural equation modeling, 100,000+ tasks	Operational
Radiology	Training of computer-aided diagnosis algorithms	Comparison of images; many tasks, much communication	In development
Radiology	Image processing and brain mapping for neuro-surgical planning research	Execution of MPI application in parallel	In development

Conclusions

- ~200 independent workers
- Enabled MTC workloads to run efficiently on Accelerators
- Enabled MIMD programmability on SIMD architecture
- Swift/T integration improves programmability
- 1,000+ Tasks/second

Future Work

- Further exploration of Intel MIC and MTC
- Enable GeMTC to utilize OpenCL and expand hardware support.
- Evaluate alternative programming platforms on GPUs (I.E., Java, Python)
- Evaluate this work on real systems and simulators



References

GeMTC – http://datasys.cs.iit.edu/projects/GeMTC
Swift - http://www.ci.uchicago.edu/swift/main/
NVIDIA - nvidia.com/object/cuda_home_new.html