Traversing a three-dimensional grid with PyCuda

Jesse Lu

December 21, 2011

1 Problem statement

We wish to efficiently traverse a three-dimensional grid of size (xx, yy, zz) in order to perform an operation at every point. However the efficiency of such a traversal is dependent upon many variables which often cannot be known or chosen, such as memory layout and chip architecture. Therefore, our strategy is to simply try multiple traversal configurations, and choose the fastest one.

We choose to use PyCuda[1] for this task, in order to take advantage of both the parrallelism of CUDA, and the metaprogramming capabilities achievable in Python.

2 Traversal options

shape (s_x, s_y, s_z) determines the size of the three-dimensional box of grid points that a single multiprocessor operates on at once.

cardinality $c = (m_1, m_2, m_3)$ determines the order of directions in which the update proceeds. Specifically, the the m_1 direction is updated first, and the m_3 direction is updated last.

3 How it works

[1] `http://mathema.tician.de/software/pycuda`