Implementation Plan

Thursday, 23 October 2014

5:37 PM

I propose to start implementation of steps 1 and 2 in parallel. Step 3, the addition of parallel execution and performance enhancement will be started once we have a reasonably stable serial implementation in Python.

If available time becomes critical, then a minimum useable parallel system on NCI would be steps 1, 2, & the course-grained task parallelism option (3.b.i), achieved via throwing some batch scripts together. One advantage of the course-grained task approach is that the Sambuca code doesn't even need awareness of the parallelism. We just run multiple jobs, one for each tile, and then stitch the results together at the end. So even if it is not the ideal final product, it might be the best approach to rapidly get a "proof of concept" running on NCI.

1. ADGC data query and preparation
   1. separate activity, to be conducted in parallel with Serial PySambuca development
   2. Constraint: Python 2 must be used in AGDC interactions
2. Serial Python Sambuca
   1. Since PySambuca is now decoupled from AGDC, could I use Python 3? This would preclude the possibility of calling into PySambuca routines directly from an AGDC pixel processor. Probably best to stick with Python 2 for now.
   2. Why?
      1. Allows my initial focus to be on getting the core Sambuca implementation right
      2. I need to understand the data and code interactions before the best way to distribute the work becomes clear.
   3. Primary references: Matlab code & interactions with the team
   4. Secondary reference: the IDL code
3. Performance enhancement (parallelisation, other)
   1. Standard approach is to have a functional serial version before optimising and scaling out across a cluster
   2. Potential approaches. Just brainstorming here, this shouldn't be decided too soon. Also, these approaches are not necessarily mutually exclusive.
      1. Course-grained task parallelism
         1. pre-processing
            1. tiles the input rasters
            2. Generates scripts to execute each tile as an independent Sambuca run

i.e.: batch system jobs on NCI/other clusters

* + - 1. Tasks submitted to cluster
      2. Post-processing
         1. Reassembles output tiles from each task into final output rasters
      3. Wrapper scripts will make it look like a single task if required (eg: job arrays)
      4. Less development effort required than the other options. The serial Python code stays the same.
    1. In-process course parallelism
       1. Integrated runs, no separate pre and post processing steps. A single task distributes the work across the cluster and takes care of the post-run data assembly
       2. Will make part of the Python code is more complex, as it includes the code to distribute work across a cluster (using MPI or the IPython parallel execution framework)
       3. But if one tile or node fails, the whole job will fail. No possibility to rerun individual tiles
    2. Fine-grained parallelism (at the pixel level)
       1. Vectorised operations via numpy/scipy (SIMD)
       2. Threading (in Python has limits)
    3. Other approaches that might help
       1. numexpr + numpy arrays (JIT compilation), might be useful
       2. Numba: too bleeding edge, difficult to use with numpy arrays
       3. Cython: code is ugly and too far from standard python. Harder for Sambuca team to maintain