

Pragmatic Concurrency for Python

See below: The OpenCL exercises of course require an OpenCL implementation. If you would like to have a go at the OpenCL exercise, A script is provided below to install the AMD/ATI CPU-based OpenCL runtime on the virtualbox, but it takes about 10-20minutes to build.

Topics covered

- Parallel programming concepts
- ipython (ipcluster)
- mpi4py (Message Passing Interface for Python)
- pyopencl

Exercises

The purpose of these exercises is not to amount to killer speed-ups (a laptop is not the right hardware for that), but rather to run and modify a few examples, become comfortable with APIs, and implement some simple parallel programs.

= mpi4py =

1) Matrix Multiplication

- Download and run the matrix multiplication examples for mpi4py, ipython and multiprocessing (a shared memory approach for Python).

matmul.tar.gz

- Configure them to have the same matrix sizes, and compare speeds.
- For ipython, you need to start an ipcluster:

```
$ ipcluster local -n 2
```

Where -n X is the number of slave processes to start.

- For mpi4py, you must start: "mpdboot". Verify it is running with "mpdtrace".

2) IPython map-reduce

Using an ipython "map" (scatter&execute) operation, get a collection of processes to count the occurrences of a word in a collection of documents, and then reduce the results to a total count.

See also: <http://en.wikipedia.org/wiki/MapReduce> [<http://en.wikipedia.org/wiki/MapReduce>], <http://labs.google.com/papers/mapreduce.html> [<http://labs.google.com/papers/mapreduce.html>]

= OpenCL =

OpenCL - Open Computing Language

"OpenCL (Open Computing Language) is the first open, royalty-free standard for general-

purpose parallel programming of heterogeneous systems. OpenCL provides a uniform programming environment for software developers to write efficient, portable code for high-performance compute servers, desktop computer systems and handheld devices using a diverse mix of multi-core CPUs, GPUs, Cell-type architectures and other parallel processors such as DSPs."

– <http://www.khronos.org/opencvl> [<http://www.khronos.org/opencvl>]

We will get comfortable with OpenCL using the AMD Multi-core CPU OpenCL implementation of OpenCL v1.1. This implementation runs on all x86 SSE2 CPUs. Installing an OpenCL driver for a recent ATI or NVIDIA GPU can result in significant (10x-100x) speedup for suitable applications.

1) OpenCL Matrix Multiply

- Install as below
- Following the `pyopencvl_f4simple_matmul.py` example included in the installation, write an `opencvl` kernel for matrix multiplication where each `global_id` computes 1 element in the output matrix.

Hint: Your "host" code will be almost identical to "`pyopencvl_f4simple_matmul.py`", you need to write a new kernel which is simpler, in that it only has to loop over the row of A and the column of B, summing as it goes and writing to the location in C determined by the thread's `global_id`.

2) OpenCL sum (reducing add)

The goal of this exercise is to optimize a reduction operation using local memory. Unfortunately, I don't think the CPU implementation has fast local memory. For those who get this far, we can try out the solutions you produce below on the GPU to show that for reduction operations, local memory give a significant performance boost.

2.1) Implement a reduction operation with the "gather funnel" architecture as presented in the lecture. Use a workgroup size=1, and write "pair answers" directly to global memory.

2.2) Implement a reduction operation which uses an optimal workgroup size for your GPU (NV=32, ATI=64), and inside this workgroup, reduces in a local memory buffer, before proceeding to a global memory reduction.

How big is the speed up? → GPU local memory allows "fine grained" parallelism.

Further reading: http://neuralensemble.org/meetings/talks/CodeJam3_Kloeckner_PyOpenCL.pdf [http://neuralensemble.org/meetings/talks/CodeJam3_Kloeckner_PyOpenCL.pdf]

= OpenCL Installation =

To get started, download the file below and save it to your vbox in its own directory (i.e. `~/opencvl`): `pyas-opencvl-cpu.tar.gz`

Change into the directory containing the file and at a terminal:

```
$ tar -zxvf pyas-opencvl-cpu.tar.gz
```

```
$ cd pyas-opencvl-cpu $ python install_amd_opencvl.py
```

This will take ~10-20mins, so go and have a tea-break :P

materials/parallel.txt · Last modified: 2010/10/07 15:33 by python-faculty

Except where otherwise noted, content on this wiki is licensed under the following license:CC Attribution-Noncommercial-Share Alike 3.0 Unported [<http://creativecommons.org/licenses/by-nc-sa/3.0/>]