Sourav Mukherjee

mobile:	**+91 8826138428**
email:	**sourav01mukherjee@gmail.com**
residence:	New Delhi

Objective

To find opportunity where I would be engaged in solving challenging algorithmic, analytical and/or software development problems everyday. Broadly speaking, I am interested in work that involves one or more of these: **1)** developing efficient algorithms, **2)** refactoring, optimizing, or scaling up existing code, **3)** writing programs to automate repetitive, manual tasks, **4)** learning high-level libraries/APIs that enhance productivity in areas like data manipulation, web app development, etc., **5)** creating insightful visualizations.

Skills

Programming languages	**Python** (6 yrs), C++ (1 yr), C (2 yrs), JavaScript (6 mths), Fortran (2 yrs), Java (1 yr), Scala (3 mths), bash/shell-scripting (1 yr)
Numerical and visualization libraries	**Numpy**, Matplotlib, Mathematica, GraphViz (for graph and network visualization)
Parallel programming APIs	CUDA for GPU programming, OpenMP, MPI
Version control	Git (https://github.com/s0urav) , Mercurial
OS	Ubuntu Linux, Windows, Android
Other software	IPython / Jupyter notebooks, GIMP, Inkscape, LibreOffice

Work Experience

National Centre for Medium Range Weather Forecasting (NCMRWF), Noida

Project Scientist 'C'

August 2015 to October 2016

Skills utilized/gained:	Python, Numpy, Matplotlib, shell-scripting, Mercurial

- Implemented algorithms to analyze and interpolate weather data in **Python** and **Numpy**. Created visualizations of the data in Matplotlib.
- Benchmarked the UM weather model on a supercomputer.

Centre for Atmospheric Sciences, Indian Institute of Technology, Delhi

(**July 2011 to July 2015**, experience listed below in reverse chronological order)

(1) Project Associate (data visualization)

July 2011 to August 2011

Skills utilized/gained:	Python, Numpy, Matplotlib, data-visualization

Used **Python**, **Numpy** and **Matplotlib** to create visualizations of weather data over the Indian subcontinent.

(2) Project Associate (porting and parallelization of numerical codes for GPUs)

September 2011 to December 2012

Skills utilized/gained:	CUDA, Fortran, parallel programming, code refactoring and porting

I worked as a **GPU programmer** in a government-funded project whose objective was to develop a GPU-accelerated, high-resolution weather model for improving monsoon predictions. My responsibility was to port the Fortran source-code of a serially implemented weather model, LMDZ (http://lmdz.lmd.jussieu.fr/), to **CUDA**. My key experiences and accomplishments were as follows:

- Manually ported several computationally intensive subroutines in the dynamics component of LMDZ to CUDA Fortran:
    - **1st attempt**: I translated the original, serial Fortran code to CUDA Fortran one `DO`-loop at a time, keeping the organization of the translated code nearly the same as the original code. It resulted in a **5x speedup**.
    - **2nd attempt**: The key factor impacting the performance of the CUDA code written in the previous attempt was the kernel call latency of too many short CUDA kernels. In order to get better speedup, I combined compatible subroutines into longer and fewer CUDA kernels, which lead to a much better **42x speedup**. However, the drawback of this code re-arrangement was that the CUDA version now looked very different to the original version, and therefore, was hard to maintain and keep in sync with any change in the latter.

- The tradeoff between the performance and the maintainability of code manually ported to CUDA prompted me to look for ways to automate the porting process. I identified several tasks in the porting process that needed to be automated, for e.g.:
    - the generation of boilerplate code, such as data-transfer statements between host and device memories based on the parameters and the return value of a subroutine,
    - the construction of a dependency graph for a sequence of subroutines so that they could be separated into independent modules that could be run in parallel,
    - the generation of programs to compare the performance of CUDA kernels and the original subroutines they are based on, etc.

In the next phase of my employment in the project, I started researching and working on these ideas.

(3) Project Scientist (automated porting and parallelization of numerical codes)

January 2013 to July 2015

Skills utlized/gained: Python, CUDA, Fortran, C++11, software porting, parser and compiler development, metaprogramming, Mercurial

I surveyed existing research and tools related to automatic parallelization but did not find anything remarkably useful. As a result, I started devising and implementing parsing, program analysis, and code generation algorithms on my own. I used **Python** for all the **algorithm development**, which greatly improved my command of the language. One of my key realizations was that Python could be used as **a metaprogramming language to specify numerical computations** (like those present in LMDZ dynamics) **at a much higher level of abstraction** than is possible in Fortran or C++. From this high-level abstraction, source-code for multiple targets could be generated, and it would also enable code refactoring and porting to be done programmatically instead of manually. This idea was partially realized when I **designed a mini-DSL in Python** for array arithmetic and array broadcasting, and implemented a compiler to translate statements written in this DSL into both **C `for`** loops and **Fortran `DO`** loops: https://github.com/s0urav/sabkuchh/tree/master/codebuilder

Seminars attended:

- Gave a presentation on my experiences with porting LMDZ at the workshop "2013 Programming weather, climate, and earth-system models on heterogeneous multi-core platforms" held at the National Center for Atmospheric Research (NCAR), USA in September 2013 (https://www2.cisl.ucar.edu/sites/default/files/mukherjee_6a.pdf).
- Gave a presentation titled, *Dynamical core of the LMDZ weather model on GPGPU* at the seminar ADCMAOC (Advanced Dynamical Core Modeling for Atmospheric and Oceanic Circulations) held at National Atmospheric Research Laboratory (https://www.narl.gov.in/), Andhra Pradesh in February 2013.

SCUBE Scientific Software Solutions Pvt. Ltd.

Technical Consultant for Wolfram Mathematica

September 2010 to June 2011

Skills utilized/gained:	Mathematica, Java, delivering technical presentations

- Delivered presentations and training courses on Mathematica at various academic, research, and corporate organizations.
- Responsible for offering clients Mathematica solutions relevant to their area of work or research and resolving technical queries.
- Co-presented in Mathematica India tours 2010 (December) and 2011 (February).
- Certified as a **Wolfram Instructor** in April, 2011. https://web.archive.org/web/20150910072400/https://www.wolfram.com/training/instructors/mukherjee.html

- Published two demonstrations:
  - http://demonstrations.wolfram.com/LyapunovFractals/
  - http://demonstrations.wolfram.com/TwoPhaseSimplexMethod/

Education

Degree / Certificate	Institute / School	C.G.P.A. / percentage
Integrated M.Sc. in Mathematics and Computing (2004 - 2010)	Indian Institute of Technology (IIT), Kharagpur	6.53 / 10
C.B.S.E (XII) (2003)	Somerville School, Vasundhara Enclave, New Delhi	79%
C.B.S.E (XI) (2001)	Somerville School, Vasundhara Enclave, New Delhi	84%

Projects

A parallel algorithm to generate Space Filling Curves

May 2016 to present

Skills utilized/gained:	Python, Numpy, Matplotlib, parallel algorithm development

Devised a non-recursive and embarrassingly **parallel algorithm** to generate the Hilbert space-filling curve (SFC). It turned out that by changing parameters, it could also generate the Z-order SFC, and many other SFCs probably unnamed in existing mathematical literature. Currently working on extending the algorithm to generate SFCs that can fill up non-rectangular grids.

Application of the Arnold Transformation to non-square images and its CUDA implementation

February 2016 to April 2016

Skills utilized/gained:	Python, Numpy, CUDA, C++11, parallel algorithm development, image processing, OpenCV

The Arnold Transformation (AT) is a one-to-one mapping between the elements of two square matrices. Practical uses of the transformation include image encryption. I extended the algorithm to work for non-square rectangular matrices, prototyped a parallel version and verified its correctness using Python and Numpy, and then rewrote the parallel version in CUDA C++, which turned out to be **tens of times faster** than the serial implementation written in C++.

Master's Thesis: Sigma Labelling of Hypercubes of order 6

September 2008 to May 2009

Skills utilized/gained:	Java, C, Python, Numpy, Matplotlib, graph theory

A "sigma labeling" is an assignment of consecutive natural numbers to the vertices of a graph, satisfying certain conditions. It **was an open problem** whether a sigma labeling exists for hypercubes of order 6 and higher (even) orders. Using a genetic algorithm, I discovered multiple sigma labelings for the hypercube of order 6. Later, I implemented a faster search algorithm based on simulated annealing.
https://github.com/s0urav/sabkuchh/tree/master/hypercube_labeling

Awards and Scholarships

- Was selected for the K.V.P.Y fellowship in 2001 and attended the workshop in Science conducted at Indian Institute of Science, Bangalore (May, 2002).
- Awarded 1st prize twice (2015, 2016) in the English-Hindi translation competition organized during the annual week-long event held at NCMRWF called "Hindi Pakhwada".