SC13 HPC Educators Program Proposal:

Strategies for Introducing Parallelism with Python

Steven Bogaerts
Department of Computer Science
DePauw University
Greencastle, IN
stevenbogaerts@hotmail.com

Joshua Stough
Department of Computer Science
Washington and Lee University
Lexington, VA
stoughj@wlu.edu

The Python programming language has seen rapid growth in popularity both in academe and industry. As a lightweight high-level language that supports both functional and object-oriented programming, it provides many tools to allow programmers to easily express their ideas. This expressiveness extends to programming using parallelism and concurrency, allowing the early introduction of these increasingly critical concepts in the computer science core curriculum. In this half-day session we describe and demonstrate an educational module on parallelism, including materials on distributed systems and parallel image processing. We cover such key concepts as speedup, divide & conquer, communication, and concurrency. We consider how these concepts may be taught in the context of CS1 and CS2, and we provide extensive hands-on demonstrations of parallelized search, sort, image processing, and distributed computing.

SC13 HPC Educators Program Proposal:

Strategies for Introducing Parallelism with Python

Steven Bogaerts
Department of Computer Science
DePauw University
Greencastle, IN
stevenbogaerts@hotmail.com

Joshua Stough
Department of Computer Science
Washington and Lee University
Lexington, VA
stoughj@wlu.edu

We are excited to propose a half-day 2013 HPC Educators session that will provide a hands-on tutorial for teaching parallel computing concepts through the Python programming language. While our module is suitable for CS 1 and 2, it will also include many principles of integration of parallelism applicable to other undergraduate courses. Having presented a related session at HPC Educators 2012, we will provide many additional concrete examples, strategies, and lessons-learned as we continue to refine our materials based on in-class experiences. We will demonstrate parallel computing models, including new applications in image processing and distributed systems. In the following we motivate the use of Python and its parallel facilities in the introductory curriculum, and we outline the materials and attendee benefits of our proposed session.

Parallelism has revolutionized how computing is done, but the corresponding revolution in how computing is *taught* remains immature, as illustrated by the fact that many curricula still relegate much examination of parallelism to upper-level algorithms and elective courses. There are a number of initiatives aimed at effecting the desired evolution in pedagogy, such as the NSF-TCPP Core Curriculum initiative on Parallel and Distributed Systems, the Intel Academic Community, and CSinParallel. However, it is striking that in these efforts Python is much less represented than other languages, even though Python is an increasingly popular introductory language. We argue that Python is a natural fit for early integration of parallelism into the computer science curriculum, and we propose to share our documented successful strategies for this in a session at the 2013 HPC Educators Program.

While Python is increasingly used for prototyping and larger scale projects, a key advantage of Python is as an introductory language with extremely simple syntax. We and other proponents of Python in introductory courses argue that this allows students to dive into interesting problem-solving much earlier than many other languages – that computer science students must *eventually* learn what Python hides (e.g. type declaration, memory management) but that these concepts need not be mastered *first*.

Given the syntactic simplicity, it is not surprising that parallel facilities in Python follow a similar philosophy. Python enables programmers to start working on concepts of parallelism quickly by hiding as many lower-level details as is practical. This enables students to gain a broad perspective of crucial concepts in parallelism, such as decomposition, processor load, communication, and deadlock, without the abundant technical and syntactic hurdles present in many other languages' parallel facilities. Thus Python provides a context in which the development of this perspective can be initiated much earlier in the curriculum than would be possible with many other parallel programming tools. With this foundation, students are well-prepared to then explore the technical and syntactic details of parallel tools in faster languages such as Pthreads or MPI in C.

Over years we have developed extensive materials and strategies for teaching parallel concepts in CS1 and CS2 that exploit Python's expressive power. We have also presented various aspects of this work at a 2012 HPC Educators session and other venues. We propose to share our approach and in particular

highlight more recently developed materials in parallel image processing and distributed systems, which provide students a deeper appreciation of the systems at work in their daily online lives. While our presentation will focus on Python, those using other languages and/or teaching other courses will learn very useful strategies that may be modified to their own work. Thus the prerequisite for the session is only some programming experience in any language.

Throughout our session, attendees will accumulate and interactively develop for themselves a sizeable collection of Python programs for their courses. These programs will demonstrate concepts ranging from introductory functional programming to the parallel programming mechanisms available in Python's multiprocessing and subprocess modules. Our goal is to have the attendees gain fluency with the materials while also periodically observing the strategies as an educator. From the session, attendees will:

- Learn about key parallel programming facilities in Python.
- Obtain many example programs, from a "Hello World" style introduction to real-world applications of parallel programming in Python, including sorting, pipelining, image processing, and distributed systems. All examples will be appropriate for CS1 and CS2 courses.
- See demonstrations of intuitive analogies and physical exercises to illustrate concepts of parallelism, in which the attendees play the role of processors and devise parallel strategies for concepts like communication and domain decomposition.
- Consider strategies for integration of these concepts into existing courses with minimal disruption to already full syllabi.

The HPC Educators sessions are an ideal forum in which to provide valuable, actionable materials to participants for teaching high-performance computing concepts. We would be excited to offer our module in this year's program.

Thank you for your consideration,

Steven A. Bogaerts
Joshua V. Stough

SC13 HPC Educators Program Proposal (Bogaerts / Stough):

Strategies for Introducing Parallelism with Python

Hands-On Materials: Demos and Exercises

This document summarizes the hands-on materials we'll use for the session. Italicized file names indicate files available now for reviewer consideration at http://cs.wlu.edu/~stough/SC13/.

All code has been tested in Linux, Windows, and Mac OS. All that is required of an attendee is a (preferably 2+ core, but not necessary) machine with command-line interface and Python 2 and IDLE (an editor that comes standard with Python). Many examples also work in Python 3.

- **Introductory Python Programming** – To give a quick introduction to the core language.
  - *pythonIntro.py* – amalgam of python code snippets demonstrating conditionals, loops, functions, lists, list comprehensions, etc., serving as a quick reference.
- **Parallel Python Programming** – We have numerous examples that will be worked in an interactive lecture and lab, covering pools, processes, pipes/queues, locks (see Section 4 of Outline).
  - *parallelHello.py* – Simple example demonstrating domain decomposition and the use of Pool/map in the multiprocessing module.
  - *parallelMontePi.py* – More advanced Pool/map example timing the sequential and parallel Monte Carlo estimation of pi. Motivating parallelized integration via Riemann Sum.
  - *spawningProcesses.py* – These examples demonstrate the basic creation of processes, including passing arguments and a customized name. The notion of a lock is necessarily introduced in order to facilitate printing.
  - *speedup.py* – These examples demonstrate how to conduct basic timing in Python. This is then applied to examine the speedup of the use of multiple processors on a problem.
- **Communication and Memory Architectures** – These are physical exercises, in which students play the roles of processors to sort a deck of cards. With a little guidance, we will quickly come to the mergesort algorithm while also considering the action of multiple processors (attendees) working in parallel; we observe speedup from working *smart* and from working *in parallel*.
  - *communication.py* – These examples demonstrate the use of a queue as a simple means to pass information between processes.
  - *parallelQuicksort.py* – example showing the use of Pipe communication (like queue). Motivating development of parallel mergesort.
- **Additional Applications**
  - *applications.py* – These examples demonstrate the application of multiprocessing to various problems, from a simple calculation of the quadratic formula to k-medoid clustering and a pipeline for floating-point addition.
- **Parallel Image Processing** – examples from making images sepia-toned and other "Photoshop" effects to applying basic computer vision techniques over many images in parallel.
- **Distributed Computing**– In the end, much distributed computing software is a simple "ssh somewhere doSomething". This code shows that fact in as few layers as practicable, with Python's multiprocessing and subprocess modules to maintain a job queue and manage jobs on remote hosts. At the top level, a list of command lines is simply generated and executed in parallel.
  - *distributedPythonCode_1.0.zip* – necessary and readme files together. Examples will be shown in genomics and computer vision.
- **Solutions to in-session worked problems** – will be available with the materials (in solutions folder), including parallel mergesort, numerical integration, sum of primes, and more.

SC13 HPC Educators Program Proposal (Bogaerts / Stough):

Strategies for Introducing Parallelism with Python

Outline of Proposed Educator Session

- Python Background (10 minutes)
  1) Installation
  2) Why Python for parallelism?
- Brief Introduction to Python Programming (30)
  1) Key programming structures
  2) LAB: Crash course on conditionals, loops, functions, lists, list comprehensions, $O(N^2)$ sorts
  3) Introduction to parallel programming technologies in Python
- Level-Appropriate Introduction to Parallel Programming Concepts (30)
  1) Experiences of integrating this material at various times in the course
  2) How to fit it in while still covering traditional course concepts
  3) Organization of materials to enable student success
  4) Lessons in parallelism with Python
     - Required programming background
     - Spawning processes in Python
     - Locks to protect shared resources
     - Cooperation between processes
- Python Pools, Processes, Pipes/Queues, Locks (45)
  1) Creating Pools, domain decomposition
  2) LAB: Pool/map
     - Parallel mergesort, numerical integration via Riemann Sum, sum of primes, Monte Carlo methods
  3) Creating Processes, interprocess communication through queues, pipes
  4) Measuring speedup
  5) LAB: Process/Pipe, and sorting
     - Parallel quicksort, mergesort
- Parallel Image Processing (40)
  1) Color enhancement and other "Photoshop" effects over many images in parallel.
- Distributed Python: examples of sequential programs solving embarrassingly parallel problems, including genomics and computer vision. Parallelism as a medium for exploring traditional course topics (15)
  1) Parameter passing
  2) Simple data structures
  3) Encapsulation
- Conclusions, acknowledgements, resources (10)

Curriculum Vitae

Steven A. Bogaerts

Department of Mathematics and Computer Science
Wittenberg University
P.O. Box 720
Springfield, OH 45501

Phone: (937) 327-7865
Fax: (937) 327-7851
Email: sbogaerts@wittenberg.edu

EDUCATION

- Ph.D. Computer Science — June 2007
  Indiana University, Bloomington, IN
- M.S. Computer Science — December 2002
  Indiana University, Bloomington, IN
- B.S. Computer Science — May 2000
  Rose-Hulman Institute of Technology, Terre Haute, IN

TEACHING

- **Positions Held**
  - Assistant Professor of Computer Science — Fall 2007 – present
    Wittenberg University, Springfield, OH
  - Course Instructor — Fall 2001, Fall 2006
    Indiana University, Bloomington, IN
  - Associate Instructor — 6 sem. in 2001 – 2007
    Indiana University, Bloomington, IN

- **Courses Taught**
  - Computing in the Arts and Sciences
  - Introduction to Programming I
  - Introduction to Programming II / Data Structures
  - Sequential and Parallel Algorithms
  - Programming Languages
  - Artificial Intelligence
  - Cybersecurity
  - Senior Seminar
  - Discrete Mathematical Structures

RELATED RESEARCH

- **Related Grants**
  - **Major Grants**
    - "Accelerators to Applications - Supercharging the Undergraduate Computer Science Curriculum", PIs: Eric Stahlberg, Melissa Smith — 2009 - 2013
      $190,250.00 received, NSF grant CCF-0915805
  - **Minor Grants**
    - Travel support from workshop organizers, to attend the Third NSF/TCPP Workshop on Parallel and Distributed Computing Education (EduPar-13) — 03/02/13
      $1,500.00 received
    - Travel support from workshop organizers, to attend the First NSF/TCPP Workshop on Parallel and Distributed Computing Education (EduPar-11) — 05/16/11
      $1,500.00 received
    - co-PI in proposal to make Wittenberg an early adopter of the NSF/TCPP Curriculum Initiative on Parallel and Distributed Computing — 01/31/11

PIs: Steven Bogaerts, Eric Stahlberg, Kyle Burke
$2,500.00 received
Plus two NVIDIA GTX480 GPUs (approx. $500 value)

- **Related Papers and Presentations**
  - Bogaerts, S. Hands-On Exploration of Parallelism for Absolute Beginners with Scratch. Proceedings of the IEEE International Parallel & Distributed Processing Symposium, Boston, MA, May 2013.
  - Bogaerts, S., Burke, K., and Stahlberg, E. Experiences in Parallel and Distributed Computing Education. Third NSF/TCPP Workshop on Parallel and Distributed Computing Education (EduPar-13), Boston, MA, May 2013.
  - Stahlberg, E., Bogaerts, S., Burke, K., and Smith, M. Parallel and Accelerated Computing Experiences for Successful Industry Careers in High-Performance Computing. Supercomputing 2012: The International Conference for High Performance Computing, Networking, Storage, and Analysis, Salt Lake City, UT, November 2012.
  - Bogaerts, S., and Stough, J. Python for Parallelism in Introductory Computer Science Education. Educators Program, Supercomputing 2012: The International Conference for High Performance Computing, Networking, Storage, and Analysis, Salt Lake City, UT, November 2012.
  - Bogaerts, S. Very Early Parallelism. Association for Computing Machinery Special Interest Group in Computer Science Education conference, Intel-sponsored session on parallelism in education, March 2012.
  - Bogaerts, S., Burke, K., and Stahlberg, E. Integrating Parallel and Distributed Computing Into Undergraduate Courses at All Levels. First NSF/TCPP Workshop on Parallel and Distributed Computing Education (EduPar-11), Anchorage, AK, May 2011.
  - Bogaerts, S., Burke, K., Shelburne, B., and Stahlberg, E. Concurrency and Parallelism as a Medium for Computer Science Concepts. Curricula for Concurrency and Parallelism workshop at Systems, Programming, Languages, and Applications: Software for Humanity 2010 (SPLASH-2010), Reno, NV, October 2010.
  - Stahlberg, E., and Bogaerts, S. Integrating Accelerated Computing into the Undergraduate Computer Science Curriculum. Symposium on Challenges, Solutions, and Visions for the Future of Computer Science Education, Franklin University, Columbus, OH, April 2009.

- **Related Student Research**
  - Copeland, P., Hildrebrandt, J., and Fink, D. A Parallel Implementation of the Minimax with Alpha-Beta Pruning Algorithm. Butler Undergraduate Research Conference, April 2012.
  - Heyder, E. Refining the Parallel Prefix Sum Algorithm. Midstates Conference for Undergraduate Research in Computer Science and Mathematics 2011 (MCURCSM 2011), November 2011.

Curriculum Vitae

Joshua V. Stough

Department of Computer Science
Washington and Lee University
204 West Washington Street, Parmly 408
Lexington, VA 24450 USA

Phone: (540) 458-8811
Fax: (540) 458-8255
Email: stoughj@wlu.edu
Web: http://cs.wlu.edu/∼stough

Education

The University of North Carolina, Chapel Hill, NC *2001-2008*
Ph.D., Computer Science 2008 (M.S. 2006)
Advisor: Stephen M. Pizer
Topic: Image analysis: Clustering and shifting of regional appearance for deformable model segmentation, with a target of Radiation Oncology.

Carleton College, Northfield, MN *1997-2001*
B.A., Computer Science and Mathematics, with Distinction
Graduated *Magna Cum Laude*, GPA 3.8

Honors and Awards

- NSF-TCPP Early Adopter: Curriculum Initiative on Parallel and Dist. Computing (*2011*)
  -Curriculum grant ($1500), expenses/poster at IPDPS 2011 ($1500)
- SuperComputing (SC) - HPC Educators Program (2010, 2011, 2012)
  -Python for Parallelism Session (2012)
  -Curriculum Resource Fair poster (2011)
- Four Lenfest summer research grants, five RE Lee student research grants (*2010-2013*)
  -Institutional funding for summer research (see Research below)
- Future Faculty Fellowship, Center for Teaching and Learning, UNC-Chapel Hill (*2006*)
- UNC Radiology Research Symposium, honorable mention, best Oral Presentation (*2004*)
- Phi Beta Kappa (*2001-Present*)

Teaching and Related Experience

- **Assistant Professor, Washington and Lee University** *2009-Present*
  In addition to CS 0/1/2 in Python, specializing in courses on image processing, computer graphics, computer vision, and artificial intelligence/machine learning.
  - CS0 Survey of Computer Science (W2012)
  - CS1 Intro. Python (2x, most recent W2013)
  - CS2/DS in Python (4x, most recent W2011)
  - Computer Graphics, C++/OpenGL (W2011)
  - Image Analysis, Matlab (2x, W2013)
  - Computer Networks, Java (W2012)
- **Visiting Assistant Professor, Claremont McKenna College** *2008-2009*
  Fall 2008: taught introductory and advanced programming (CS 1 and 2, respectively) in Java. Spring 2009: taught CS 1 and an image analysis course of my design that is influenced by my experience in the field.

- **Instructor, UNC-Chapel Hill** *Spring 2005, Fall 2007*
  Developed and taught an introductory programming course in Java. Bore full responsibility for lectures, assignments, examinations and grading for the classes of ∼35 students each.
- **Teaching Assistant, UNC-Chapel Hill** *Spring 2002*
  Assisted in a computer organization course, grading assignments and exams, preparing study sessions, and tutoring individually on digital logic design and assembly languages for a class of 30.
- **Guest Instructor, UNC-Chapel Hill** *2004-2007*
  Designed and conducted occasional lectures in graduate courses on image analysis and computer vision.
- **Tutor, Chapel Hill-Carrboro City Schools, NC** *2005-2007*
  - Tutored high school students in all subjects, though predominantly algebra and geometry, for the 21st Century Community Learning Center Program.
  - Tutored middle school students in all subjects under the Blue Ribbon Mentor - Advocate Program.
- **Undergraduate Positions, Carleton College** *1998-2001*
  Math Skills Center tutor in Calculus, Statistics and Computer Science
  Teaching Assistant for Calculus I

Research Interests

- Parallel and Distributed Computing education
- Image appearance models for medical image segmentation
  -ISBI reviewer 2009-Present
- Visual object recognition, machine vision
- Scientific Visualization
- Object-oriented data analysis, including clustering and statistical modeling
- Deformable shape model techniques

Selected Publications

- **Stough, J.V.**, Ye, C., Ying, S., Prince, J.: Thalamic parcellation from multi-modal data using random forest learning. In: *International Symposium on Biomedical Imaging (ISBI)* 2013, http://ieeexplore.ieee.org/.
- **Stough, J.V.**, Greer, L., Benson, M.: Texture and Color Distribution-Based Classification of Live Coral. International Coral Reef Symposium (2012).
- **Stough, J.V.**: Clustering and shifting of regional appearance for deformable model segmentation. PhD Dissertation (Advisor: Stephen M. Pizer). University of North Carolina at Chapel Hill, 2008. See http://midag.cs.unc.edu/.
- **Stough, J.V.**, Broadhurst, R.E., Pizer, S.M., and Chaney, E.L.: Regional appearance in deformable model segmentation. In: *Information Processing in Medical Imaging (IPMI)* 2007, Springer LNCS 4584, 532-543.
- **Stough, J.**, Pizer, S.M., Chaney, E.L., and Rao, M.: Clustering on image boundary regions for deformable model segmentation. In: ISBI 2004, http://ieeexplore.ieee.org/.

SC13 HPC Educators Program Proposal (Bogaerts / Stough):

Strategies for Introducing Parallelism with Python

Release Statement

To the SC13 HPC Educators Program organizers,

Steven Bogaerts and Joshua Stough agree to the release of our session materials on USB memory stick, SC13 website, or other formats for the SC13 HPC Educators Program.

Signed,

Steven Bogaerts
Joshua Stough

SC13 HPC Educators Program Proposal (Bogaerts / Stough):

Strategies for Introducing Parallelism with Python

Travel Support Request

We are each requesting travel support for attendance at SC13. As per the requirement of Educators Program attendees, we intend to participate during the entirety of SC13.

Joshua Stough requests:

- SC13 Education and Technical Programs conference registration fees waived
- Round-trip airfare purchased through SC travel system
- Conference-paid hotel room for six nights for a shared double room
- Meal stipends for six days.

Steven Bogaerts requests:

- SC13 Education and Technical Programs conference registration fees waived
- Round-trip airfare purchased through SC travel system
- Conference-paid hotel room for three nights for a single occupancy room
  (Will pay for 3 additional nights with some home university support)
- Meal stipends for six days.