I. Course Description

Data-intensive computing deals with storage models, application architectures, middleware, and programming models and tools for large-scale data analytics. In particular we study approaches that address challenges in managing and utilizing ultra-scale data and the methods for transforming voluminous datasets (big data) into discoveries and intelligence for human understanding and decision making. Topics include: storage requirements of big data, organization of big data repositories such as Google File System (GFS), characteristics of Write-Once-Read-Many (WORM) data, semantic organization of data, data-intensive programming models such as MapReduce, fault-tolerance, privacy, security and performance, services-based cloud computing middleware, intelligence discovery methods, and scalable analytics and visualization. This course has four major goals: (i) understand **data-intensive computing**, that has been defined as *the fourth paradigm* for Sciences by the late Jim Grey, (ii) study, design and develop solutions using ***data-intensive computing models*** such as *MapReduce*, (iii) predictive analytics using packages such as R and Google analytics and (iv) focus on methods for scalability using *the* ***cloud computing infrastructures*** such as Google App Engine (GAE), Amazon Elastic Compute Cloud (EC2), and Windows Azure.

II. Course outline: On completion of this course students will be able to analyze, design, and implement effective solutions for data-intensive applications with very large scale data sets. More specifically a student will be able to:

1. Describe data-intensive computing concepts
2. Compute with data-intensive computing concepts
3. Recognize a data-intensive problem.
4. Identify the scale of data.
5. Analyze the data requirements of a problem.
6. Describe the data layout and define the data repository format (Ex: GFS, Hive)
7. Decide the algorithms (Ex: MapReduce) and programming models (Ex: Bayesian)
8. Define application-specific algorithms and analytics (Ex: Finite State Element analysis)
9. Design the data-intensive program solution and system configuration.
10. Implement the data-intensive solution and test the solution for functional correctness and non-functional requirements.
11. Research algorithmic improvements and experiment with them.
12. Write a report summarizing the solution and results.
13. Incorporate services from cloud computing platforms.
14. Study the foundational concepts enabling cloud computing: services-based interface, programmatic consumption of services, virtualization, PKI-based security, large-scale storage, load-balancing, machine images and on-demand services.
15. Formulate data-intensive visualization solutions for presenting the results.

III. Course Information

Website:	http://www.cse.buffalo.edu/~bina/cse487/spring2013
Instructor:	Bina Ramamurthy (bina@buffalo.edu)
Lecture Time:	TTH: 3.30-5.00pm
Lecture Location:	200G Baldy
Office Hours:	MWF: 10.10-11.10AM
Office:	345 Davis Hall

IV. Grading (tentative) Overall grade for the course will be based on the student's performance in: class attendance and participation (5%), 2 exams (40%), 3 projects (45%), presentation/demo (10%)

V. Text Book There are two text books that cover the two of the major goals defined in the description (data-intensive computing: fourth paradigm, data-intensive computing models, cloud computing) respectively:

1. The Fourth Paradigm: Data-Intensive Scientific Discovery**,** Tony Hey (editor), Stewart Tansley (editor) and Kristin Tolle(editor), Microsoft Research (October 16, 2009), ISBN-10: 0982544200, ISBN-13: 978-0982544204, **online version is available at**: http://research.microsoft.com/en-us/collaboration/fourthparadigm/default.aspx

2. Data-Intensive Text Processing with MapReduce, Jimmy Lin and Chris Dyer, Synthesis Lectures on Human Language Technologies, 2010, Vol. 3, No. 1, Pages 1-177, (doi: 10.2200/S00274ED1V01Y201006HLT007). **An online version of this text** is also available through UB Libraries since UB subscribes to Morgan and Claypool Publishers. Online version available at: http://lintool.github.com/MapReduceAlgorithms/index.html

For the other goals of the course we will use online material/publications:
3. Predictive analytics: Google Prediction API :
https://developers.google.com/prediction/docs/getting-started

4. Cloud infrastructures: we will focus on amazon for the infrastructure (though Windows Azure and Google App Engine are equally good): http://aws.amazon.com/documentation/

VI. Tentative Schedule (will be provided later)

VII. Project Plans
Each project will involve complete installation of all the necessary toolkits, software packages and servers by each student (or group of students) in their workspace. Students will also write a detailed technical report on the project they implement. Students can work in groups of no more than 2 people. Choose the group members with complementary expertise.

VIII. Suggested Reading List

1. K.A. Delic and M.A. Walker. Emergence of the Academic Computing Cloud. ACM Ubiquity, *Ubiquity Volume 9, Issue 31 (August 5 - 11, 2008), http://www.acm.org/ubiquity/volume_9/v9i31_delic.html, last viewed January 2009.*
2. J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004, 137-150;
3. Jeffrey Dean, Sanjay Ghemawat: MapReduce: a flexible data processing tool. Commun. ACM 53(1): 72-77 (2010).
4. J. Gray. Distributed Computing Economies. Computer Systems Theory, Technology and Applications, From Object-Relational Mappers Vol. 6, No. 3 - May/June 2008, http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=545, last viewed August 2011.
5. The Hadoop Project. http://hadoop.apache.org/, last visited August 2011.
6. M. Johnson, R. H. Liao, A. Rasmussen, R. Sridharan, D. Garcia and B.K. Harvey. Infusing Parallelism into Introductory Computer Science Curriculum using MapReduce, EECS Department, University of California, Berkeley, April 2008, http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-34.html, Technical Report: UCB/EECS-2008-34.
7. Google Code University: Material for Students and Educators: http://code.google.com/edu/parallel/index.html

IX. Course Policies

Attendance Policy: You are responsible for the contents of all lectures and recitations (your assigned section). If you know that you are going to miss a lecture or a recitation, have a reliable friend take notes for you. Of course, there is no excuse for missing due dates or exam days. We do, however, reserve the right to take attendance in both lecture and recitation. We may use this information to determine how to resolve borderline grades at the end of the course, especially if we see a lack of attendance and participation during lecture sessions. During lectures, we will be covering material from the textbook. We will also work out several of the problems from the text. Lecture will also consist of the exploration of several real world problems not covered in the book. You will be given a reading assignment at the end of each lecture for the next class.

Incomplete Policy: We only grant incompletes in this course under the direst of circumstances. By definition, an incomplete is warranted if the student is capable of completing the course satisfactorily, but some traumatic event has interfered with their capability to finish within the timeframe of the semester. Incompletes are not designed as stalling tactic to defer a poor performance in a class.

Academic Integrity Policy: UB's definition of Academic Integrity in part is, "Students are responsible for the honest completion and representation of their work". It is required as part of this course that you read and understand the departmental academic integrity policy located at the following URL:
http://www.cse.buffalo.edu/undergrad/policy_academic.php
There is a very fine line separating conversation pertaining to concepts and academic dishonesty. You are allowed to converse about general concepts, but in no way are you allowed

to share code or have one person do the work for others. You must abide by the UB and Departmental Academic Integrity policy at all times. **NOTE:** Remember that items taken from the Internet are also covered by the academic integrity policy! If you are unsure if a particular action violates the academic integrity policy, assume that it does until you receive clarification from the instructor. *We reserve the right to check or question any portion of any work submitted at any time during the semester or afterwards.* If you are caught violating the academic integrity policy, you will minimally receive a ZERO in the course.

Exams Policy: There will be a midterm (Exam 1) that will be administered and graded before the resign date. Midterm material will cover all lecture and reading assignments before the exam, as well as concepts from the project assignments. Midterms are closed book, closed notes, and closed neighbor. The second exam (Exam 2) will be covering all lecture material after exam1 and all the projects. We do not give make up exams for any reason. If you miss an exam, you will receive a zero for that portion of the grade. Second exam will be on the last day of classes.

Students with Disabilities:

If you have special needs due to a disability, you must be registered with the Office of Disability Services (ODS). If you are registered with ODS please let your instructors know about this so that they can make special arrangements for you.