CS546 DCS Miniproject

Assigned: 10/5/15. Due: 1/6/15. For Rx, seminars between 23-31/5/15.

Work in teams of 2 on one of Px or Rx. Teams must divide the work between the members. However, each team member must be able to explain the complete mini-project. In your report, document the contribution of each member. Besides common demo/presentation, there will be individual vivas.

Write your report in your own words, being careful to avoid plagiarism. See http://www.-plagiarism.org. All reports will be checked for plagiarism using Turnitin. *Plagiarism will be strictly penalised.*

Programming Problems

Make your design concurrent and scalable using multiple threads and processes. Test your programs with small datasets where you can verify the correctness. Document your testing. All measurements must include confidence intervals at 95% confidence level, and must scale up to multiple PCs.

All programs *must* follow a coding standard. Submit your programs, results, analysis and conclusions, and demonstrate execution.

P1. A shared calendar for use by a boss and secretaries. The calendar resides on a server. The users are usually connected to the server but need to have local copies of the calendar while travelling. When reconnected, the local calendar and shared calendar need to be made consistent. In case of conflicts that cannot be automatically reconciled, alert the users for manual reconciliation. Use the `.ics` format, and support import/export from/to a `.ics` file.

P2. Design and implement a simple call rating engine for postpaid telephony. Generate fictitious calls. At the end of each call, compute the balance of the user via the rating engine, then update the subscriber's balance with the accounting engine. Test for millions of active subscribers and 100s of concurrent calls.

P3. Design and implement a mirror engine that creates a local copy of a website. The design should allow a variable number of concurrent crawlers, N_c. To avoid overloading the website, the mirror engine should restrict the rate of download to a specified maximum. Measure the performance of mirroring Insite and Moodle with N_c = 1, 2, … Compare the actual download throughput with the specified maximum.

P4. Choose a non-trivial problem in engineering or science that makes significant use of matrices. Write an MPI program to solve this problem. Measure the performance of the program, varying the number of processes, number of cores, etc.

P5. Design and implement a simple online lending library. Support addition of books to the inventory, issue and return of books. The borrowing charge for each book is 10% of the price of the book. An Account Server maintains the balance of each member. Before issuing a book, verify with the Account Server that the balance is sufficient to cover the charge. Test for 10s of millions of books, millions of members, and 100s of simultaneous transactions.

P6. Design and implement a network discovery engine. Given 1 or more IP address subnets, the discovery engine finds all IPs addresses that are in use. It uses ping to every IP address in the given subnets. Your program should be able to discover 10.8.13.* in 1 minute and 10.8.*.* in 2 hours. Be careful to not overload the network.

P7. It is desired to create an index of word occurrence in short messages (SMs).

a. Write a script to create a corpus of a large of SMs. Each SM consists of an SM id, and up to 140 characters of text. Choose text from some corpus of online documents such as man pages, selecting only alpha characters and space. SMs need not be correct or meaningful sentences.

b. Divide your corpus into N_D equal partitions and store each partition in the local disk of each of N_D "data store" PCs.

c. Write a MapReduce program to generate the index. The result should be stored in a control PC.

d. Evaluate the performance of your indexer. Use corpus size of up to 100m SMs and $N_D = 1$ to 20.

P8. It is desired to find the number of degrees of separation between a popular individual and her admirers in a social network.

a. Generate a social network randomly where a link between nodes A and B indicate friendship (one degree of separation).

b. Write a MapReduce program to compute the degrees of separation using Djikstra's algorithm.

c. Evaluate the performance of your program using large social networks with millions of nodes. Use 1-20 PCs.

Research Problems

For each research topic, read at least 2 papers and/or textbook chapters. Papers must be from IEEE/ACM journals or conferences, or the attached list. Textbook chapters must be ones not covered in the course. Do some original work based on what you've read. *40% of the credit is for the original work.*

Each team must write a report of at least 3,000 words on the topic and make a presentation in class. Presentations must be made with OpenOffice Impress or with LaTeX. Use of PowerPoint is not permitted.

R1. Clock synchronisation using NTP

R2. Clock synchronisation using PTP

R3. The CAP Theorem

R4. Reliable broadcast in distributed systems

R5. Testing concurrent systems

R6. The Apache Spark cluster computing framework

R7. The end-to-end argument in systems design

Suggested References

This list contains some references for the mini-project research topics. These are not mandatory, you are free to select your own list of references.

NTP and PTP

1. Mills, David L. "Internet time synchronization: the network time protocol", *IEEE Trans. Commun.* 39.10 (1991): 1482-1493.
2. Sethi, Adarshpal S., Hongxiang Gao, and David Mills. "Management of the Network Time Protocol (NTP) with SNMP." *Computer and Information Sciences Report* (1997): 98-09.
3. Mills, David L. "A brief history of NTP time: Memoirs of an Internet timekeeper." *ACM SIGCOMM Computer Communication Review* 33.2 (2003): 9-21.
4. G.V. Neville-Neil, "Time is an Illusion", *CACM*, Jan. 2016, pp. 50-55.

Reliable Broadcast

1. Kaashoek, M.F., A.S. Tanenbaum, and S.F. Hummel. "An efficient reliable broadcast protocol." *ACM SIGOPS Operating Systems Review* 23.4 (1989): 5-19.
2. Bellur, Bhargav, and Richard G. Ogier. "A reliable, efficient topology broadcast protocol for dynamic networks." *INFOCOM '99. Proc. 18th Ann. Joint Conf. IEEE Computer and Communications Societies, IEEE*. Vol. 1. IEEE, 1999.

Parallel Languages and Compilers

1. Foster, I.T., and K.M. Chandy. "FORTRAN M-A Language for Modular Parallel Programming." *Journal of Parallel and Distributed Computing* 26.1 (1995): 24-35.

Computation Models for Big Data

1. Karloff, Howard, Siddharth Suri, and Sergei Vassilvitskii. "A model of computation for MapReduce." *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2010.
2. Goel, Ashish, and Kamesh Munagala. "Complexity measures for map-reduce, and comparison to parallel computing." *arXiv preprint arXiv:1211.6526,* (2012).
3. Wada, K. "Computational Models for Big Data Processing", 2nd *IEEE Int'l Symp. Computing and Networking (CANDAR), 2014*.

Testing Concurrent Systems

1. P. Maddox, "Testing a Distributed System", *CACM*, Sep. 2015, pp.54-58.

Distributed Storage Systems

1. J. Ousterhout, "The Future of Volatile Storage", *IEEE Spectrum*, Nov. 15.

Distributed Systems Design

1. E. Brewer, "CAP twelve years later: How the "rules" have changed," *IEEE Computer*, vol. 45, no. 2, Feb. 2012, pp. 23-29.
2. J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-To-End Arguments in System Design", *ACM Trans. on Computer Systems*, v 2, n 4, Nov 1984, pp 277-288.