CS550 "Advanced Operating Systems"

Project Proposal Handout

Assigned: August 30th, 2017

Due: September 6th, 2017 by 12:00am (midnight)

Overview

A major component of your grade will be based on a project. The topic of the project will be chosen by the student(s) and approved by the instructor. The project proposal (no more than 2 pages excluding the title page and references) is **due by Sept. 6, 2017**. Based on the project ideas listed in the course website, you are to choose one of these topics, or develop one by yourself, and expand it into a proposal. Your proposal should include the following information:

- Title
- people involved (in case it is a group project no more than 3)
- Motivation and background information
- Problem statement
- Related work
- Proposed solution
    - clearly state the nature of the project (e.g. implementation of a real system, simulation, theoretical, empirical performance evaluation, survey, etc)
    - be specific about what techniques you plan to use, what existing software and systems you will use, etc
- Evaluation: be specific with what metrics you plan to examine
- Timeline with weekly goals
- Deliverables
- Conclusions
    - what do you think you will learn (keep it short)
    - be specific with what how you will evaluate if your project is a success
    - work partition (if more than one person involved)
- References

You should use single spaced 10pt font, with 1" margins. Feel free to contact me if you have any questions. Also, you should utilize my office hours as well as those from the TA to meet with us in person and talk about your project. Please submit your proposal write-up via blackboard, by the deadline stated at the top of this assignment.

Here we list some possible project topics. You can talk to the instructor and the TA about your idea and thought, to pick one of the topics or develop one by yourself.

Performance tuning of Hadoop over OrangeFS

OrangeFS is an open source Parallel File System (PFS). It was designed based on Parallel Virtual File System (PVFS). It was designed for High Performance Computing (HPC) environments such as MPI-based scientific simulations. Recently it added support for Hadoop to allow Hadoop applications running directly on top of OrangeFS instead of the native Hadoop Distributed File System (HDFS). How it performs compared to the HDFS is very interesting to explore the possibility of unified storage system for both environments. In this project, you need to run standard Hadoop benchmarks with OrangeFS as storage layer and explore the optimal configuration of Hadoop and OrangeFS for the best performance.

Spark-based data analytics on large scale simulation result

In big data era, Spark is becoming a popular framework for data analytics, machine learning and graph processing. Spark usually works on large dataset generated from other applications or collected from widely spread sensors. As one of the data producer, large scale scientific simulations generate enormous data for both fault tolerance and data analysis. Spark could be a capable paradigm to enable scalable data analysis on large scale simulation result data. In this project, you need to write a survey of Spark-based data analytics on large scale simulation result and explore one pair of simulation application and Spark-based analytics code (performance test, scalability test).

An Empirical Evaluation of Distributed Key/Value Storage Systems

In this project, you are going to evaluate various distributed key/value storage systems.
You must choose 3 of the following 10 distributed key/value stores to evaluate:

- Amazon DynamoDB (https://aws.amazon.com/dynamodb/)
- MongoDB (https://www.mongodb.org)
- Cassandra (http://cassandra.apache.org)
- CouchDB (http://couchdb.apache.org)
- HBase (http://hbase.apache.org)
- Riak (http://basho.com/products/#riak)
- ZHT (http://datasys.cs.iit.edu/projects/ZHT/)
- Redis (http://redis.io)
- HyperTable (http://www.hypertable.com)
- Oracle NoSQL Database (http://www.oracle.com/technetwork/database/databasetechnologies/nosqldb/overview/index.html)

On each instance/node, a client-server pair is deployed. Test workload is a set of key-value pairs where the key is 10 bytes and value is 90 bytes. Clients sequentially send all of the key-value pairs through a client API for insert, then lookup, and then remove. Your keys should be randomly generated, which will produce an All-to-All communication pattern, with the same number of servers and clients. The metrics you will measure and report are:

- Latency: Latency presents the time per operation (insert/lookup/remove) taken from a request to be submitted from a client to a response to be received by the client, measured in milliseconds (ms). Note that the latency consists of round trip network communication, system processing, and storage access time

- Throughput: The number of operations (insert/lookup/remove) the system can handle over some

period of time, measured in Kilo Ops/s.

Make the necessary plots to visualize your data. Explain why your results make sense; what explicit things did you do to verify that your performance as you expected?

Replicated File Distributed System

This file system should provide transparent replication. In particular, you can suppose there are n copies of a data file and n server modules. Each server provides access to one copy of the file. A client interacts with any one of the server modules. The servers interact with each other to present clients with the illusion that there is a single copy of the file. Each file exports four operations: open, close, read and write for client. The file servers interact with each other to ensure that copies of the file are kept consistent and that at most one client at a time is permitted to write into the file. Each file servers has a local lock manager process that implements a solution to the readers/writer problem. Some points to bear in mind.

a. Each fileserver exports two sets of operations: those called by its clients and those called by other file servers. Each server module keeps track of current access mode by using a lock manager. For example, the file is not written if it was opened for reading. Multiple readings however are allowed. If a certain operation needs to get permission from all lock managers, please acquire the locks in the same order for all clients. Otherwise, deadlock may occur.

b. Within write procedure, a module first update its local copy then concurrently updates every remote copy. It is analogous to using a write-through cache update policy. An alternative approach would be to update the remote copy when the file is closed.

Demo purpose: Store duplicated simple text files in several fileservers for demo purpose. Write operation just simply appends more text to the file. Client gets connected to any fileserver in a graphical user interface.

Parallelizing an Application

You may parallelize an application that you already have experience with (assuming that a desired parallel solution to the application doesn't yet exist). You may use any of the parallelization schemes you learned in this course (threads, MPI, OpenMP, or MapReduce/Hadoop). Your grade will be determined based on the appropriateness of your parallel design, the difficulty of the parallelization, the resulted performance, and the depth of your performance analysis.

Implementing A Distributed System Protocol

There are several distributed system protocols, including totally ordered broadcasts, distributed snapshots, and Paxos distributed consensus. You will implement a distributed system protocol and devise your own test cases to examine the usefulness, correctness, and performance of your implementation. Note that we will not provide test cases or test scenarios. It is your responsibility of devising them and the quality of your test cases will be an important basis to determine your grade.

Resource Management for Multicore / Hyperthreaded Processors

Multicore / hyperthreaded processors share hardware resources at fine-grained levels. However, contention from conflicting resource requirements (especially in the cache / memory hierarchy) can result in reduced performance. You might already have had such experience in your earlier programming assignments. In this project, you will attempt to identify one or more mechanisms by

which conflicts between applications running on sibling hyperthreads / cores may be detected. You can also implement a scheme to utilize the information to improve performance. There have been a large body of research efforts devoted to this topic. It may be helpful to read some previous works. You can start by checking these papers and their references:
https://192.5.53.208/u/sandhya/papers/eurosys09.pdf
http://jcst.ict.ac.cn:8080/jcst/CN/article/downloadArticleFile.do?attachType=PDF&id=9732
http://static.usenix.org/events/usenix09/tech/full_papers/zhang/zhang_html/
https://www.usenix.org/system/files/conference/atc15/atc15-paper-srikanthan.pdf

Nonblocking Data Structures

These are thread-safe concurrent data structures that work without locks, and are immune to the performance hiccups caused by preemption. Possible projects include

- Dual data structures: A few years back, researchers have rewriten classic queues, stacks, synchronous queues, and exchangers as lock-free "dual" data structures, in which an operation that has to wait for a precondition leaves an explicit reservation in the data structure. What else can be built in this style? In particular, you might consider priority queues or other search structures.
- Hybrid transactional / non-blocking data structures: One of the key advantages of nonblocking data structures is increased concurrency. A major disadvantage is complexity. Can we get (most of) the concurrency without (most of) the complexity by using transactions for sub-parts of each operation? Consider, for example, a B-tree or 2-3 tree: can we capture rebalancing as a series of transactions, each of which transforms the tree from a consistent but sub-optimal state to a "better" consistent state?

Accelerating K means on GPU with CUDA programming.

Focus is on the basic usage of the cuda language, the exploitation of the most important features of the device (massive parallel computation, shared memory, texture memory) and efficient usage of the hardware to maximize performance.

A Survey on Concurrency Control for Distributed Database Systems

Concurrency control is one of the major issues in database systems; therefore, many concurrency control algorithms based on different strategies have been proposed. The problem for centralized database management systems is well understood. Distributed concurrency control, by contrast, is still in a state of turbulence. The survey will research the proposed algorithms, and investigate for general solution, or summarize the current development.

A Survey on the Load Balancing Aspect of Distributed Systems

You should write a survey on the load balancing aspect of distributed systems. Heterogeneous networks involve processors with different computing power, different memory capacity, or they have different communication links. How do they diffuse and balance the load? The survey will research to identify the problems in load balancing and summarize the progress of on-going development.

Other Potential Topics for Projects :

1. Grid Computing
2. Parallel Compilers
3. Contemporary Languages/Libraries for Parallel Computing
4. GPU Computing
5. Dataflow Computing
6. Cloud Computing

7. Parallel/distributed data-mining (Netflix Prize: http://www.netflixprize.com/)
8. Static and dynamic task partitioning and data distribution strategies for parallel computing
9. Distributed Algorithms for Mobile Sensor Networks
10. Parallel game programs (e.g. chess, othello, go, etc.)
11. Design and implementation of parallel algorithms for the solution of nontrivial combinatorial optimization problems (e.g. Traveling Salesman, 0/1 Knapsack, Graph Partitioning, 8-Quinn's, etc.).
12. Compute intensive applications in science and engineering that require supercomputing power for solution (computer simulations of natural phenomena in physics, chemistry and engineering).
13. Parallel image processing algorithms (parallel volume rendering/ray tracing, segmentation, region growing, etc.)
14. Future Trends in Massively Parallel Computing