Course Project Ideas

Below are some ideas for starting points for CS87 course projects. You are welcome to use any of these or come up with an idea of our own. Since this is a course project it should be a substantial piece of work. Also, it should have something to do with parallel or distributed computing, and it must have some component that looks like science; there should be a general question that you are trying to answer though your project.

I have not thoroughly investigated how feasible it is to implement my suggestions below. As a result, all of these project ideas will require some background investigative work and will require further definition of the problem you are solving and how you plan to solve it. You should start with a general idea of the problem you want to look at, then do a literature search of related work and projects to get some ideas of what has been done and what some approaches are to solving your problem. It is fine to do a project whose goal is to reproduce other researcher's results, and/or to compare different, already developed, approaches to a particular problem. However, there should be some question that you are trying to answer through your project.

I strongly encourage you to work in groups of 2 or 3 on course projects. I think you will get much more out of the experience if you do so.

1. **Evaluate a parallel or distributed algorithm or class of algorithms, or develop a new parallel algorithm**

   This project could be based on a performance and scalability study of different known algorithms, or it could be based on developing a new parallel solution to a problem. Both should have an experimental component, but the former is largely experimental.

   You could compare a set of algorithms implemented in different ways (compare some of MPI, OpenMP, CUDA, Hadoop, pthreads, RPC, ...). Run and compare on different systems. What are the limitations of different systems? of different implementations?

   For a project that examines existing algorithms, it should have a large experimental and analysis phase, such as a detailed scalability analysis of different implementations including analysis to discover why and when one version is better than another (strengths and weaknesses). Perform enough analysis to characterize and explain the application's performance and scalability for the different algorithms and/or paradigms, and to explain why the performance is different (or not different) for the different algorithms/paradigms...what are the parts of the execution time that are causing the application to perform as it does. How do communication patterns and frequency change as program changes size? What is the major contributing factor that determines the program's performance and does this change as the program/machine size change? What conclusions can you draw from this study?

   Examples:

   (a) Look at example applications in the case study section of Part 1 of "Designing and Building Parallel Programs", by Ian Foster, Argone National labs (available on-line: http://www.mcs.anl.gov/ itf/dbpp/).

   (b) sorting algorithms: look at the same algorithm in different languages, or different sorting algorithms in a smaller number of languages. There is a sorting competition (see http://sortbenchmark.org/), you could try to come up with a good algorithm for one or more of the categories.

   (c) A game playing algorithms that involve a large search space. Don't try chess, it is too big.

   (d) Approximation algorithms for NP complete problems like TSP. You could look at heuristics that use branch and bound or simulated annealing.

   (e) Look at some of the Teragrid applications to use as a starting point

   (f) Look at Distributed computing projects that benefit humanity (http://www.hyper.net/dc-howto.html) for some ideas.

(g) Look at an algorithm you have used in another class, and examine ways to parallelize it. Some suggestions: parallel join in databases, parallelize machine learning, parallelize information retrieval algorithms, parallelize data mining applications, parallelize computational biology, chemistry, physics, astronomy, ..., applications. Basically, think about problems that process large amounts of data and/or require large amounts of computation, and look into parallel solutions for these problem.

2. **Improve part of MPI or optimize for clusters of SMPs**

   You could focus on the performance of a particular MPI communication routine to optimize for a particular target. There has been a lot of work in this area, so look at optimizing a collective communication for a particular target system. You may be able to use MRNet (available on-line: http://www.paradyn.org/mrnet/).

   Another idea would be to look at incorporating openMP, pthreads, or shared memory segments (see man page for shmget) into openMPI for SMPs or clusters of SMPs.

   You could look at using threads and shared memory in some way, maybe process spawns on the same machine would use threads instead of processes? The clone system call on linux may be of help You could modify the openMP implementation, or you could look at modifications at the language or compiler level.

3. **GPUs**

   Look at implementing better system support for general parallel computing on GPUs. What does openMP, MPI, CUDA support for GPUs look like right now, and how could it be improved?

   The recent SC conference proceedings have a lot of papers in this area:
   http://sc08.supercomputing.org/scyourway/conference/Paper.html
   http://supercomputing.org/

   Search for GPU or graphic processors or CUDA, and you will find lots of stuff. The idea is to look at projects that use GPUs for general parallel computing.

   Here are a few older references:

   (a) Zhe Fan, Feng Qiu, Arie Kaufman, Suzanne Yoakum-Stover. GPU Cluster for High Performance Computing. In proceedings of SC '04.

   (b) Kayvon Fatahalian, Jeremy Sugerman, and Pat Hanrahan. Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication. In Graphics Hardware 2004.

4. **Smart ssh lab.cs** In our lab, if you ssh into lab.cs, you are logged into some lab machine. Currently, we use a round-robin algorithm to pick which lab machine. Come up with a better algorithm based on system usage information. The idea would be to avoid heavily loaded machines in favor of idle or lightly loaded machines.

   The decision should not take long, which means you will likely need a daemon process running on each node that will report usage information to participants.

   You could think about a centralized or decentralized solutions

   /proc will likely be of great use

5. **Distributed Job Placement**

   This project is similar to the ssh lab.cs project, except it is a batch job placement system. You could also think about using mpirun as the job distribution system that you will make smarter.

   Implement smart remote process placement on the lab machines (or on a cluster) perhaps with support for heterogeneity. A program will be submitted on one node in the distributed system (or the head node on the cluster) and your system will choose the best node(s) on which to run. Your system will

pick the "best node(s)" for the job based on information about each nodes' capabilities, resources, and current resource load.

If you do this for a cluster, you should never pick the head node as a node to run the job, but if you do this for the lab machines, you could pick the submit node as the node on which the program will run.

You should implement a script that a user will run to to trigger your system to choose the best location for the process to run (see ssh for man page for help with spawning a remote process).

Your system should have immediately available information about other node's capabilities and usage, so that it can immediately choose the "best" node. This means that you likely want some daemon processes running continually on every node, that exchange info about themselves.

You should try and test different policies for picking "the best" placement for a job, and use some benchmark programs to run experiments to compare them.

6. **Job scheduling in clusters or clusters of SMPs**

   Job scheduling in parallel, distributed, and cluster systems is a huge research area. You could examine and implement and test different scheduling policies that you read about for a cluster (or the lab machines). And, perhaps you can come up with your own solution There may be a parallel scheduling simulator that you can use to test different solutions. Often these run on traces from real systems. You could consider all parallel or mixed parallel and sequential workloads.

7. **Peer-to-Peer Systems**

   There are a lot of interesting questions related to P2P. Some have to do with the type of file being distributed, others have to do with availability, or searching, or scalability, or consistency. Here are a couple more specific ideas:

   (a) Peer-to-Peer Backup.

   The idea here is to write a distributed back-up facility for all/part of the file system. You could start with existing peer-to-peer software, and add support for distributed back up of the file system, or you could start with a simple client-server application that you use as a guide for how to modify a P2P system later. One suggestion is to try backing up all or part of the /local directories on some of the lab machines. /local is not NSF mounted (each node's /local contents differ), and we do not currently do backups of /local, so your complete, correct, efficient and robust final project could actually be used in our lab.

   Your solutions should assume that all peers will participate in creating and storing parts of the backup (don't just copy the contents of one machine's /local completely to another peer, but break it up in some way based on each node's capacity). You should think about reliability issues (what if a machine dies, can you restore its /local from backup). When a peer wants to do a backup (you can think of this that is something that is done once every night, for example), it should search for peers with available storage space, and transmit part of the back up to the set of peers. You also need to make sure that you can restore files from the backup that is spread out over some number of peers. Your solution should ensure that P2P backups don't completely fill up each node's disk space after few backups (you may want to think about limiting the number of backups and/or backing up only diffs after the first backup).

   (b) Peer-to-Peer Consistency Specification.

   Think about adding consistency specification to a P2P system. The idea would be that the owner of a file could specify different consistency constraints associated with updates to the file. Strong consistency, for example, might mean that a write is visible at all other copies instantly "atomically", weak consistency, might mean that updates get propagated eventually to other copies (it is up to you to define these). As an entire project this is way too big, but you could think about tackling a piece of it such as implementing a way to one type of consistency (pick one that allows multiple writers though). You could use an existing P2P system (bittorrent might be a good choice) or you could simulate a P2P system to test your ideas.

8. **Finding idle RAM in a cluster**

   OS's sometimes hide idle RAM in system caches (these are OS caches not HW caches like L1, L2 and L3). As applications run and files are opened and closed, pages of RAM get allocated to different system caches. Over time, the system can look as if there is not much idle RAM, when in fact there is a lot of RAM allocated to cache space that is not really being used.

   This project would continue an investigation started by Joel Tolliver'10 on how to identify idle RAM in Linux clusters and then use this information to develop heuristics for when this idle RAM can be allocated for other use and when it should be release by this other use back to the system.

   This is project related to my research project Nswap. Heuristics would be used to trigger growing and shrinking of the Nswap Cache space. It also has applications to scheduling in general purpose clusters.

9. **Ideas for cluster projects**

   (a) **A cluster-wide monitoring tool**

   Implement a cluster performance monitoring tool (like vmstat, top, fstat for cluster-wide global resources). It should run on the head node, get cluster-wide information, and present it in an easy to understand way.

   You will likely need a global naming scheme (look at Mosix, OSCAR, or SSI as a reference systems).

   (b) **A reliability testing tool for clusters**

   Design and implement a test bed system for testing and monitoring a cluster. For example, add support for turning on/off different types of failures (bad disk blocks, bad device, bad node, network disconnects, ...) to allow for testing of reliability support of cluster software. Basically, you implement a bunch of "knobs" that you can tune to different values, each of which will trigger different types of "failures" at different rates. You don't really want to implement real failures but virtual ones; simulating a node crash shouldn't really bring that node down. Again, I'd suggest starting with SSI, OSCAR, or mosix

   (c) **SSI**

   Try implementing some feature of a SSI on one of our clusters. Start with SSI or Mosix (their project homepages may have some ideas of desired missing functionality that you can try to add)

   (d) **Cluster scalability**

   Investigate one or more cluster scalability issues: what happens when the cluster has 5,000, 10,000, or more nodes?

   ```
   reliability issues: something is almost always broken
   cluster maintenance: booting, software install and upgrades
   problems with fixed-sized kernel data structures not scaling: pids, sockets, etc.
   ```

10. **Implement reliable java RMI (Remote Method Invocation...like RPC)**

    RMI applications are written as two separate programs: the client and the server program. The server creates remote objects. Clients obtain remote references to the server's remote objects, and invoke methods on them. RMI implements the underlying communication and message passing between the client and server processes. The server is a single point of failure; if the server dies all the clients fail. Add support for replication of remote objects at a group of servers to support reliable service.

    I'm not sure what the current support for reliability is in RMI, so if this is already supported, then evaluate different ideas for implementing reliable RMI and evaluate them for space and time efficiency.

    Here is one reference that may be a good starting point:

    http://www.springerlink.com/content/knxt9axkujxur8nq/

11. Java Thread Migration (I think this will be a very difficult project).

    Add support for migrating a single thread from one Java VM to another running on a different host. You may want to look at using the Reflection class and/or JVMDI interface To do this you will need to add support for obtaining a thread's runtime state at any point in the execution, check-pointing the state, stopping the thread on one machine, and re-starting the thread on another machine from check-pointed state. You can consider a simplified case first, where there is one thread in the application so that the migrated VM doesn't need to keep executing after the thread has been migrated.

    This project may require adding methods to the Reflection and/or JVMDI interfaces, however, before you decide to make these kinds of changes you should make sure that what you need to do can't be done with some combination of interface methods.

12. **Ideas for Web pre-fetching and caching**

    (a) Develop (and implement) algorithms to pre-fetch web documents so that latency is reduced.
    Here is some interesting work on Web Mining that may give you some ideas

    `http://www.cs.washington.edu/homes/map/adaptive/download.html`

    (b) Caching of the Web documents differs from file caching in distributed file systems in some very important ways. For example, consistency constraints for web pages and files differ.

    One idea for a project is to perform a comparative study of proposed caching policies for the web and possibly proposing a new caching policy.

    Do a google search to help you find Web benchmark programs, and/or traces of Web traffic.

    There has been a lot of work on web caching, so if you do something here, you could either compare different existing approaches, or you could try to come up with a new idea.

    (c) Investigate Techniques for prefetching and caching content-specific data. Some types of WWW data may have different access patterns than other (google earth vs. standard webpages vs. ...)

13. Reliable computing: look at questions associated with continuing to function in the presence of failure. This usually means a single node fails, a server node fails, or a network becomes disconnected temporarily.