Wiki topics for Chapters 3 and 4

Please write about one of these topics on the wiki. Near the top of your wiki page, please insert

- a link to the writeup of your topic (this page);
- a link to the page (or pages) you started with, so that your reviewers can see what value you have added.

The *title* of your wiki page should be

`CSC/ECE 506 Spring 2015/3x yy` or

`CSC/ECE 506 Spring 2015/4x yy`

where "`3x`" or "`4x`" is your topic number (either `1b`, `2a`, or `2c`); and
`yy` is an arbitrary two-character string of your choosing.
That is, once you get to the wiki, in the URL in the address bar, you should replace "`Main_Page`" with

`CSC/ECE 506 Spring 2015/3x yy`

3a. Language extensions for parallel programming

Start with a brief introduction on how the three parallel programming models we have considered influence the style of programming. Then take as examples

- OpenMP for shared memory,
- MPI for message passing, and
- CUDA for data-parallel programming.

Show how each programming-language extension realizes the concepts that are characteristic of the programming model it implements. It is better to explain the different features of each programming-language extension with a coding example, rather than explaining the different directives in detail. Summarize the advantages and disadvantages of the various language extensions. You may cover *more* than these three extensions if you wish, but don't leave out any of the three.

Note: You can use the APIs/ framework /set of compiler directives in place of languages. Examples of other programming languages/API/ framework /compiler directives are OpenCL and OpenACC.

We suggest that you consult (not copy from!) the following sources.

[1] http://en.wikipedia.org/wiki/List_of_concurrent_and_parallel_programming_languages

[2] https://computing.llnl.gov/tutorials/parallel_comp/#Models

3b. Map-reduce

Begin by copying, not editing(!) the existing wiki page at
http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_506_Spring_2013/3b_xz
MapReduce is a programming model for processing massive data on large-scale clusters. It was initially proposed by Google in 2004. As indicated by the name “MapReduce”, this programming model consists of two major steps: map and reduce. In the map step, the problem being solved are divided into a series of sub-problems and distributed to different workers. After collecting results from workers, the computation enters the reduce step to combine and produce the final result. Nowadays many mainstream programming languages have their own implementations of MapReduce libraries. MapReduce implementations can work on multiple kinds of systems, not only on clusters, but also multicore and multiprocessor systems. Survey several MapReduce implementations. Identify the differences between the implementations on clusters and multicore/multiprocessor systems.

The existing page does a good job of describing MapReduce itself, but doesn’t do a very good job of describing how it’s used. There are a few examples in the “More Examples” section down at the bottom. But they are examples of problems that *could* be done with MapReduce; there is no suggestion of how to partition them. It would be good to have such an explanation for two or three examples, including a short piece of pseudocode.

Another important topic that is totally missing is what parallel architectures MapReduce can be used on. Is there an advantage to running certain kinds of algorithms on shared-memory multiprocessors? Message-passing multiprocessors? And, what about LANs?

The organization of this page has been improved from earlier versions, but problems still remain. Clearly, Google’s MapReduce and Hadoop are important. I’m not sure that Phoenix is so important, and there’s no real explanation of why it is given so much space. Is it the implementation of choice for shared-memory systems? If not, can you describe shared-memory implementations more generically, maybe using Phoenix as an example? It’s probably good to say something about MapReduce on graphics processors, but again, is Mars the best example of a GPU implementation? And why should the section on graphics processors be so much longer than the others? It would certainly be good to cut it, in favor of more useful content.

What I’d like to see is this kind of organization:

- What kinds of problems can be tackled by MapReduce on …
    - shared-memory machines;
    - distributed-memory machines;
    - LANs.
- For each of these 3 kinds of architectures, describe MapReduce implementations

generically, perhaps using a representative system as an example.

3c. Steps in parallel programming

As part of the course lecture, you have studied parallelizing ocean currents. Start with a different example from the literature, and explain the various steps involved in parallel programming like task creation, determining variable scope, task synchronization, grouping tasks into threads, mapping threads onto processors etc. Explain each step with a detailed diagram with code/pseudo code.

4b. Parallel implementations of Gaussian elimination

Begin by copying the existing wiki page at http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_506_Spring_2011/ch4a_bm . The current wiki page has a lot of code review explaining a few basic implementations of Gaussian elimination. Please take time to make the sections more conforming throughout the wiki. Also, the overview of gaussian elimination could use more details. For a few of the examples, there are detailed walkthrough on how the code works. Please extend this to all examples, as well as giving a detailed holistic description of each code segment. Lastly, try to find scholarly articles that describe new approaches to Gaussian elimination implementation.