**CSE 443/543: High Performance Computing**

**Exam #2 Study Guide**

|  |
| --- |
| **Thursday, November 11th 2021**  **During class time** |

## Text Coverage:

* *Introduction to Parallel Computing* (Access via: [On Campus](http://proquest.safaribooksonline.com/0201648652?tocview=true)  [Off campus](http://proquest.safaribooksonline.com.proxy.lib.muohio.edu/0201648652?tocview=true) links):
  + Chapters 2 (Sections 2.1 to 2.4.5)
  + Chapter 5 (Sections 5.1, 5.2, 5.3, 5.4.1, 5.4.2, and 5.6)
  + Chapter 7 (Sections 7.1, 7.2, 7.9, and 7.10)
* "C++ How to Program" by Paul Deitel and Harvey Deitel. Prentice Hall. (ISBN-10: 0-13-266236-1, ISBN-13: 978-0-13-266236-9). [Oncampus link to Safari E-book](http://proquest.safaribooksonline.com/9780134448930), [Off-campus link to Safari E-book](http://proquest.safaribooksonline.com.proxy.lib.miamioh.edu/9780134448930).

## Types of Questions to expect:

Combination of multiple-choice, fill in the blanks, short answer, “what does the following code do”, and “write the C++ code to do the following" questions. You will need to be able to read and write C++ code. You will be asked to write various C++ programs similar to those in exercises and homework assignments. The exam will include mathematical and analytical problems similar to those covered in exercises and homeworks.

## Available Materials:

Closed book, simple calculators (cannot be a graphing calculator, have a qwerty keyboard, cannot have a touchscreen, or be integrated with other devices like cell phone, iPOD etc.). Other than a calculator, no other electronic devices will be permitted during the exam. You will be provided with the selected method list (see attached document) along with your exam for your reference.

## Concepts you should know:

* **Generic Concepts**: Units of memory sizes (KB, MB, GB), Time (sec, milliseconds (msec), microseconds (usec), nanoseconds (nsec)), CPU speeds (Hz, MHZ, GHZ), FLOPS and conversion between related family of units.
* **HPDC Platforms:** Terminology and concepts covered in Chapter 2 of the book including:
  + Cache memory use and effects of caching (cache hit vs. cache miss). Locality of reference and issues with false sharing and multithreading.
  + Flynn's Taxonomy (SISD, SIMD, MISD, MIMD) in the context of Multithreading/OpenMP.
  + Differences between shared memory and distributed memory architectures. Their advantages and drawbacks.
  + Concepts of parallelism -- data parallel vs. task parallel applications. Examples.
  + Spectrum of parallelism -- approaches to developing parallel programs.
* **Parallel programming with OpenMP**:
  + Basics of threads, thread model. Processes vs. threads, motivation for using threads. Disadvantages of explicit threads.
  + Shared and non-shared thread resources.
  + Data parallelism vs. task parallelism.
  + Limitations of OpenMP, Fork-Join model,
  + OpenMP and extent rules for directives. OpenMP directives – shared, private, firstprivate, if, num\_threads, reduction, copyin.
  + OpenMP reduction clause and applications
  + OpenMP parallel directive and various clauses
  + OpenMP critical directive and critical sections
  + OpenMP for directive and various clauses
  + Scheduling strategies – static, dynamic, guided, and runtime.
  + OpenMP sections directive.
  + Race conditions, critical sections, and open MP atomic directive.
* **Analytical Modeling of Parallel Programs (10-15%)**
  + Concepts of asymptotic behavior representation (big O notation)
  + Identifying behavioral characteristics from performance graphs
  + Performance metrics for parallel systems
    - Source of overhead
    - Relationship between execution time, total parallel overhead, speedup, efficiency, and cost
    - Effect of granularity on performance, scaling characteristics of parallel programs, Isoefficiency metric for scalability, and asymptotic analysis of parallel programs.
    - You should be able to perform various quantitative and asymptotic analyses to determine the aforementioned metrics and apply them for comparing and contrasting parallel and serial programs.
* **Performance analysis of Parallel Programs**
  + Identifying behavioral characteristics from performance data
  + Performance metrics for parallel systems
    - Relationship between user time, elapsed time, %CPU and speedup (S) of multithreaded program
  + Identifying and troubleshooting performance issues in multithreaded programs
    - You should be able to identify fragments of code that degrade performance and rewrite them to eliminate performance bottlenecks.
  + **Timing measurement and analysis:** Difference between user time, system time, and elapsed time. Importance of %CPU for multithreaded programs.
    - Why timings for programs vary and why multiple runs are needed.
    - Interpreting average timing, and 95% CI, and T-test p-values.
    - Interpreting CI and comparing time.
* **Concepts of efficiency:** Runtime/CPU efficiency, memory efficiency, energy efficiency. Identifying efficiency constraints in application. Rewriting programs to balance efficiency requirements – example – rewrite method to reduce runtime while using more memory and vise versa.
* **SLURM scripts:**

1. Reading SLURM script.
2. Identify key settings, including: job name, number of cores requested, peak memory requested, wall time requested.
3. Modifying/completing a given SLURM script to perform specific operations.

* **Benchmarking and Profiling:** Concept of benchmarking and types of benchmarks. Issues involved in designing micro-benchmarks. Assessing validity of micro-benchmarks using profilers. Concept of profiling and types of profilers. Using Linux perf. Interpreting results from Linux perf.
* **C++ programming:**
  1. **Basic program constructs**
     1. Variables & expressions
     2. if-else and if-else statements
     3. switch statement
     4. Looping constructs (for, while, do-while, range-for)
     5. Functions/methods
        1. Pass by value versus pass by reference
        2. Performance and memory impact of pass-by-value
        3. Using const keyword for parameters.
     6. Default values for parameters
  2. **Classes and objects**
     1. Using std::string
        1. Constructors for string.
        2. Methods for operating and accessing strings
        3. Conversion to-and-from numeric data types to std::string.
  3. **Arrays**
     1. Basics of arrays.
     2. 1-D arrays
     3. 2-D arrays
        1. Row major organization
     4. Command-line arguments
  4. **Vectors, iterators, and algorithms**
     1. Use of vectors and iterators for processing collection of data
     2. Create type aliases via the using clause in C++
        1. Creating aliases given English description
        2. Tracing aliases back to their original types.
     3. Operations on a vector: adding elements, accessing elements, removing elements, etc.
     4. Standard algorithms such as: for\_each, copy, copy\_if, copy\_unique, sort, find, min\_element, max\_element, min, max.
     5. Using external functions with algorithms
     6. Using lambdas with algorithms.
     7. Reading/printing/writing vectors to I/O streams
  5. **Hash maps (unordered\_map)**
     1. Use of unordered\_map
     2. Using unordered\_map as associative arrays
     3. Defining and using unordered\_maps of different data types
     4. Looking-up values in unordered\_maps
     5. Iterating over all the entries in a map and processing them
  6. **Basic text file I/O operations**
     1. Reading and writing data to console using std::cin and std::cout.
     2. Using stream-insertion (<<) and stream-extraction (>>) operators to read and write data.
        1. Understanding these operators and how they handle whitespaces.
     3. Using std::getline method to read a full line of text
     4. Using std::ifstream and std::ofstream to read/write text files.
     5. Using std::istringstream and std::ostringstream to perform I/O with strings.
  7. **General programming concepts**
     1. Fundamentals of problem solving
     2. Source code, pseudo code, algorithm
     3. Syntax errors and troubleshooting them
     4. Semantic errors and troubleshooting them
     5. Functional testing (using diff to compare outputs)
     6. Debugging, debugger, break points, stack trace
     7. Concept of data types and information that can be inferred from data types
     8. Basics of files: path, absolute vs. relative path, directory vs. file. Executable vs. source file.
     9. Performance profiling and using profilers
     10. Interpreting output of profilers to draw conclusions
  8. **Other exercises**
     1. Converting English statements to corresponding C++ statements
     2. Describing C++ statements in English
     3. Code walkthroughs to determine operation and output from a C++ program
     4. Developing a C++ program given a functional description
     5. Identifying performance or memory issues in C++ programs
     6. Rewriting C++ program to address memory or performance issue
     7. Interpreting data in the form of a chart/graph
     8. Interpreting data in the form of a table and computing runtime statistics.
* **Basic Linux Commands:** basic operations on Linux, changing directory, creating programs, compiling and executing programs, observing processes, listing files, redirection of I/O at the shell.
* **Compiler optimization & libraries:** Basics of compiler optimization. Flags to use for GCC, including: -g, -Wall, -fopenmp, -O2, -O3.

## Preparation Suggestions:

1. Do read the text book materials while paying attention to implementation/application details.
2. Redo lab exercises. Develop short programs to test/verify your understanding of references. Review arrays, 2-D arrays, C++ class, constructors, how to call a constructor, the move constructor, operator overloading.
3. Review all the exercises.
4. **Review the functionality of pertinent methods and commands in the supplied method/command sheet.**
5. **Review the handouts material**.