جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Course Syllabus: High Performance Computing - AMCS 312

Division	Computer, Electrical and Mathematical Sciences & Engineering
Course Number	AMCS 312
Course Title	High Performance Computing
Academic Semester	Fall
Academic Year	2018/2019
Semester Start Date	08/26/2018
Semester End Date	12/11/2018
Class Schedule (Days & Time)	02:30 PM - 04:00 PM | Sun Wed

Instructor(s)

Name	Email	Phone	Office Location	Office Hours
David Elliot Keyes	david.keyes@kaust.edu.sa	+966128080324	0116, 1, Al-Khawarizmi (bldg. 1)	4-5pm Sundays and 4-5pm Mondays

Teaching Assistant(s)

Name	Email
Mohammed Al Farhan	mohammed.farhan@kaust.edu.sa

Course Information

Comprehensive Course Description	Prerequisites: High performance computing algorithms and software technology, with an emphasis on using distributed memory systems for scientific computing. Theoretical and practically achievable performance for processors, memory system, and network, for large-scale scientific applications. The state-of-the-art and the future promise of predictive computational science and engineering. Algorithmic kernels common to linear and nonlinear algebraic systems, partial differential equations, integral equations, particle methods, optimization, and statistics. Computer architecture and the stresses put on scientific applications and their underlying mathematical algorithms by emerging architecture. State-of-the-art discretization techniques, solver libraries, and execution frameworks. AMCS 312 is an introduction to the concepts, the hardware and software environments, and selected algorithms and applications of parallel scientific computing, with an emphasis on tightly coupled computations that are capable of scaling to thousands of processors and well beyond. It ranges from high-level descriptions of motivating applications to low-level details of implementation, in order to expose the algorithmic kernels and the shifting balances of computation and communication between them. The course is partly theoretical and partly practical, with students running demonstration codes on KAUST's Shaheen system. Modest programming assignments based on the demonstration codes using MPI and PETSc culminate in an independent project leading to an in-class report. A good subtitle of this course would be *"Algorithms as if Architecture mattered."* Actually, architecture does matter today. During decades of progress using the paradigm of bulk synchronous processing on tightly coupled systems, architecture could largely be abstracted away through the message passing interface (MPI), an excellent example of "separation of concerns" in computer science. One could write in a high-level language without concern about where the compiler and runtime stashed the operands, because flops were relatively slow, which made everything else, including the physical layout of the architecture, appear nearly flat. One could count flops for serial complexity estimation, and determine how many could be done concurrently (between synchronization events) for parallel complexity estimation. Today, however, flops are cheap compared to the cost of moving data, in both time and energy expenditure. Therefore, we must worry about the topology of the network and the memory hierarchy and the latencies and bandwidths of every part of the memory system and network in getting the operands to the FPUs. This gives high performance computing an emphasis different from some other types of computing. The same architecture advances that make it frustrating also make it exciting.
Course Description from Program Guide	High performance computing algorithms and software technology, with an emphasis on using distributed memory systems for scientific computing. Theoretical and practically achievable performance for processors, memory system, and network, for large-scale scientific applications. The state-of-the-art and promise of predictive computational science and engineering. Algorithmic kernels common to linear and nonlinear algebraic systems, partial differential equations, integral equations, particle methods, optimization, and statistics. Computer architecture and the stresses put on scientific applications and their underlying mathematical algorithms by emerging architecture. State-of-the-art discretization techniques, solver libraries, and execution frameworks.
Goals and Objectives	The overall goal of 312 is to acquaint students who anticipate doing independent work that may benefit from large-scale simulation with current hardware, software tools, practices, and trends in parallel scientific computing, and to provide an opportunity to build and execute sample parallel codes. The software employed in course examples is freely available. The course is also designed to make students intelligent consumers and critics of parallel scientific computing literature and conferences. Much of the motivation for parallel scientific computing comes from simulations based on discretizations of partial differential equations (PDEs, typically described with sparse matrices), or integral equations (IEs, typically described with dense matrices), or based on interacting particles (unstructured interaction lists, often embedded in octtrees). An understanding of the underlying equations, their physical meaning, and their mathematical analysis is important in some parts of the course and opens up many possibilities for independent projects. Other material is easily abstracted away from its underlying operator equation context to that of a generic bulk-synchronous computation that interleaves flows of data with operations on that data. The intention is to provide a course of interest to a broad clientele of graduate researchers, with the core clientele being those interested in PDE-based simulations. Students from mechanical engineering, electrical engineering, chemical engineering, materials science, and geophysics should find it of interest, in addition to computer scientists and applied mathematicians. Thirteen algorithmic prototypes that occur regularly in scientific computing have been identified in a famous 2006 Berkeley technical report "The Landscape of Parallel Computing Research: The View from Berkeley" (UCB/EECS-2006-183). Students may want to download and devour this report. The Berkeley prototypes are: dense direct solvers, sparse direct solvers, spectral methods, N-body methods, structured grids / iterative solvers, unstructured grids / iterative solvers, Monte Carlo (including "MapReduce"), combinatorial logic, graph traversal, graphical models, finite state machines, dynamic programming, backtrack/branch-and-bound. The first seven are essentially floating point kernels and the last six essential integer kernels. We will examine a few of these kernels in detail. Lecture coverage includes (not in chronological order, but interlayered for good pedagogical progression): -Introduction to large-scale simulations -Introduction to parallel architecture and programming models -Introduction to MPI, PETSc, and other software frameworks for HPC -Parallel algorithms for the solution of large, sparse linear systems and nonlinear systems with large, sparse Jacobians -Parallel algorithms for partial differential equations -Parallel algorithms for N-body particle dynamics

Required Knowledge	Systems modeled by partial differential equations and linear algebraic equations motivate this course and make the parallel applications relevant. However, it is not essential to be fluent in the mathematical analysis of such systems to become expert in their computational analysis. *The aspects of these subjects that are important to success in this course have to do with understanding the choreography of data and hardware*. If you are comfortable with following the algorithms without a theoretical understanding of how they approximate the real world (modeling) or how rapidly they converge to it (analysis), you can survive this course and even excel in it. The results of these other topics can be taken as "black boxes", just as algorithms can be taken as "black boxes" to scientists and engineers who need to compute. Complaints that the course is “too mathematical” sometimes arise from computer scientists, but are inappropriate for researchers in high performance scientific computing. Complaints that the course relies too much on computer achitecture sometimes arise from mathematicians, but are inappropriate for similar reasons. High performance computing arose because of the demands for high performance of computational science and engineering (CS&E) challenges rooted in mathematical models. It now overlaps with numerous other domains, but CS&E is at its core. Since essentially all high performance computers run some form of the Unix operating system at the user interface, a working knowledge of rudimentary Unix commands and navigation of a Unix file system must be owned in advance or picked up.
Reference Texts	None of these texts are “required,” but are of reference interest. Most are out of date in technical details, due to the rapid evolution of the field, but the principles are mostly timeless. Approximately 25 original source technical papers will be distributed during the course to support extensions of lecture material. All lectures slides will be distributed in pdf handout form and constitute the most accurate accompanying reading. -High Performance Computing, by T. Sterling, M. Anderson and M. Brodowicz (Elsevier, 2018) -PETSc for Partial Differential Equations, by E. Beuler (https://github.com/bueler/p4pdes/releases distribution, 2017) -Introduction to High Performance Scientific Computing, by V. Eijkhout et al. (Creative Commons, 2015) -Introduction to High Performance Computing for Scientists and Engineers, by G. Hager and G. Wellein (CRC Press, 2011) -Applied Parallel Computing, by Y. Deng (World Scientific, 2011) -Petascale Computing: Algorithms and Applications, by D. Bader, ed. (Chapman and Hall, 2008) -Scientific Parallel Computing, by L. R. Scott et al. (Princeton, 2005) -An Introduction to Parallel Computing: Design and Analysis of Algorithms,2nd, by A. Grama et al. (Pearson Addison Wesley, 2003) -Sourcebook of Parallel Computing by J. Dongarra et al., eds. (Morgan-Kaufmann, 2002) -Using MPI; Portable Parallel Programming with the Message-Passing Interface, by W. D. Gropp et al. (MIT Press, 1999) -Parallel Computer Architecture: A Hardware/Software Approach by D. E. Culler et al. (Morgan Kaufmann, 1999) -High Performance Computing, K. Dowd and C. Severance (O’Reilly, 1998)
Method of evaluation	**10.00**% - Active participation **30.00**% - Homework /Assignments **30.00**% - Final exam **30.00**% - Course Project(s)
Nature of the assignments	Evaluation for AMCS/CS 312 consists of four components: problem sets, project, final exam, and class participation.
Course Policies	Evaluation for AMCS/CS 312 consists of four components: problem sets, project, final exam, and class participation. Each component will have an associated numerical score, from which a course composite will be constructed. Each of the first three components will weigh approximately 30% in the composite and participation will account for approximately 10%. The implication of the weights is that one can pass the course without solid attendance, but a straight A is essentially out of reach without consistent attendance. However, one or two weeks of absence will not damage the attendance score. Problem sets may be undertaken cooperatively (and this is encouraged), but each registered student must submit the homework separately under their own name, vouching for their own responsibility for the answers. The quality of the write-up is part of the grade. It is intended that all students should be able to score well on the problem sets because they will be announced well in advance of their due dates; students will have time to seek help in office hours, in their own reading and research of the topics, and in consultations with one another. The problem sets should create an extended ongoing discussion for the class community. The project is intended to be individual. If students want to team to undertake a “bigger” project and earn the same grade for it, this should be negotiated when projects are launched in midcourse. Projects will be submitted in report form, and each project will be featured in a short presentation to the class at the end of the semester. The final exam is, of course, individual and will be administered during the time announced by the Registrar during the KAUST final exam period. Students should not travel from campus before taking the final exam, or it is not possible to pass the course within the scope of the fall semester. Students who are planning to graduate in December and who depend upon completing this course during the Fall need to budget their time throughout the semester so that their project is completed when grades are due.

Additional Information	AMCS 312 has been produced as a video course in the Blue Waters HPC course offerings for the University of Illinois at Urbana-Champaign, under the US National Science Foundation. It was offered in remote asynchronous video format on 12 campuses in the USA in Fall 2016. The same lecture material will be available in video format to KAUST students in Fall 2018, and will comprise the bulk of the "lecturing." Class time will be "flipped" with the instructor asking the students questions designed to probe comprehension, and with students asking the instructor clarifying questions. Since students are expected to view and respond to approximately 150 minutes of video lectures per week, live class time will be reduced accordingly, but students are asked to reserve the regular class hours on Sunday and Wednesday for use on an as needed basis. The syllabus of AMCS 312 in Fall 2018 will be supplemented by seminars and guest lecturers on HPC topics of their expertise. We anticipate that the same video lectures of AMCS 312 will also be offered at Aramco in Fall 2018, so there could be a non-academic credit peer group of auditors with whom to augment the technical discussions.

Tentative Course Schedule

(Time, topic/emphasis & resources)

Week	Lectures	Topic
1	Sun 08/26/2018 Wed 08/29/2018	**Unit 1: Introduction to Parallelism** **Sunday:** **Part a:** Case Study: How Many Computers Dance on the Head of a Pin? **Wednesday:** **Part b :** Parallel Programming Paradigms
2	Sun 09/02/2018 Wed 09/05/2018	**Sunday:** **Unit 2: The Versatile Laplacian:** Graph and Grid Laplacians: Applications and Discretizations **Wednesday:** **Unit 3: Introduction to Computational Science & Engineering** Context and Content of CS&E in HPC
3	Sun 09/09/2018 Wed 09/12/2018	**Eid Vacation**
4	Sun 09/16/2018 Wed 09/19/2018	**Unit4: Algorithms for Structured and Unstructured Grids** **Sunday:** **Part a:** Stencil Evaluation, Matrix Multiplication, and Krylov Methods **Wednesday:** **Part b:** Preconditioning and KrylovPSchwarz
5	Sun 09/23/2018 Wed 09/26/2018	**Sunday:** **Unit 5:Introduction to PETSc** **Wednesday:** **Unit 6: Nonlinear Root Finding** **Part a**:Newton and Continuation Methods
6	Sun 09/30/2018 Wed 10/03/2018	**Sunday:** **Vacation** **Wednesday:** **Unit 6: Nonlinear Root Finding** **Part b:**NewtonPKrylovPSchwarz
7	Sun 10/07/2018 Wed 10/10/2018	**Sunday:** **unit6 : Nonlinear Root Finding** **partc:**Case study: External Aerodynamics (Gordon Bell Prize, 1999) **Wednesday:** **Unit 7: Fast Multipole Methods** **Part a:** FMM Algorithm and Analysis
8	Sun 10/14/2018 Wed 10/17/2018	**Sunday:** **Unit 7: Fast Multipole Methods** **Part b:** FMM in Parallel and Generalizations **Wednesday:** **Unit 8: Direct Methods for Dense Linear Systems**

9	Sun 10/21/2018 Wed 10/24/2018	**Sunday:** **Unit 8: Direct Methods for Dense Linear Systems** **Wednesday:** **Unit 9: Direct Methods for Sparse Linear Systems**
10	Sun 10/28/2018 Wed 10/31/2018	**Sunday:** **Unit 9: Direct Methods for Sparse Linear Systems** **Wednesday: Computing exercises** Computing Problem #1 review and Computing Problem #2 prospect
11	Sun 11/04/2018 Wed 11/07/2018	- **Unit 10:Algorithmic Adaptations to Extreme Scale**
12	Sun 11/11/2018 Wed 11/14/2018	**Unit 11:Multigrid Methods** **Sunday:** **Part a:**Introduction to Multigrid **Wednesday:** **Part b:** Multigrid beyond Poisson
13	Sun 11/18/2018 Wed 11/21/2018	**Unit 12:Parallel Communication** **Sunday:** **Part a:** Message Passing Models **Wednesday:** **Part b:** Hybrid Programming Models
14	Sun 11/25/2018 Wed 11/28/2018	**Sunday:** **Unit 12:Parallel Communication** **Part c:** Communication Performance Modeling **Wednesday:** **Unit 13: Partitioning for Parallelism**
15	Sun 12/02/2018 Wed 12/05/2018	**Unit 14: Case Studies** **Sunday:** **Part a:** Accelerated Cyclic Reduction **Wednesday:** **Part b:** Spectral Methods in CFD
16	Sun 12/09/2018	**Sunday:** **Unit 14: Case Studies** **Part c:** Benchmarking **Wednesday:** **Course Review**
17		- **Project Reports**
18		

Note

The instructor reserves the right to make changes to this syllabus as necessary.