Carnegie Mellon University
Master of
Software Engineering

17-647: Data Intensive Scalable Systems

TR 3:05 - 4:25pm ET
Course Zoom Link:
https://cmu.zoom.us/j/99729815711?pwd=a0hLUnRrUnlwM1NrVlBkYlZXTEJldz09
Meeting ID: 997 2981 5711
Passcode: 460686
[A4, Spring 2022, 6 Units]

Last Update: 2/14/22

Instructor	Email	Office location & hours
Matthew Bass	mbass@andrew.cmu.edu	SCR 267 Zoom ID: 3609899473 By appointment
Paulo Merson	pmerson@andrew.cmu.edu	Zoom ID: 7741902826 By appointment
Teaching assistants		
Riyaz Panjwani	rpanjwan@andrew.cmu.edu	Tue 4:30 - 5:30pm
Lakshay Virmani	lvirmani@andrew.cmu.edu	Fri 11:30am - 12:30pm

Course Description. Internet services companies such as Google, Yahoo!, Amazon, and Facebook, have pioneered systems that have achieved unprecedented scale while still providing high level availability and a high cost-performance. These systems differ from mainstream high performance systems in fundamental ways. They are data intensive rather than compute intensive. They are also cloud-native distributed systems that are often deployed as microservices and require integration with external systems via networked-based APIs. They need to inherently support scalability, typically having high throughput, security, reliability and availability demands as well.
Designing and building these systems require a specialized set of skills. This course will cover design principles and strategies needed to design and implement data intensive scalable distributed systems. In this domain engineers not only need to know how to architect systems that are inherently scalable, but to do so in a way that also supports high availability, reliability, and performance. Given the large distributed nature of these systems basic distributed

systems concepts such as consistency and time and synchronization are also important. These systems largely operate around the clock, have components running on different nodes and platforms, so the course places an emphasis on deployability and operational concerns.
The course includes a hands-on project where students get to create and discuss design alternatives, and implement the solutions in a public cloud environment. The basic concepts will be given during the lectures and applied in the project. The students will gain exposure to the core concepts needed to design and build such systems as well as current technologies in this space. Class size will be limited.

Learning Objectives

Students will be introduced to concepts, design principles and patterns for distributed systems in general, with a focus on cloud-native http services, asynchronous event-driven communication, and SQL and NoSQL data stores.. Students will be expected to demonstrate the application of this knowledge through the construction of systems with increasing expectations. Students in this class will learn what it takes to engineer systemic properties into data intensive distributed systems. Namely, we will focus on Interoperability, scalability, availability and reliability, loose coupling, modifiability, and distributed data. Students will learn about the interplay among these concerns when applying several design patterns and technologies. Managing tradeoffs appropriately is often one of the most difficult engineering challenges and will be a key concern in different stages of the project.

Prior Knowledge. Students are expected to be familiar with programming in at least one programming language suitable for creating backend services, such as node JS, Java, and C#. Formal training in Software Architecture is helpful, but not required.

Students should have some experience writing small programs or software applications. Students in doubt regarding their experience should obtain instructor's permission.

Learning Resources. The course and all course materials will be distributed online and accessible with a CMU account via Canvas.

Assessments. Students learn more by applying and explaining ideas to others, thus, the course requires the following activities:

- **Lecture and Reading Assessments:** These are short online questions derived from the required readings and lectures
- **Class participation:** There are exercises students will do in recitations in pairs or small groups to practice applying the concepts learned in the course as well as participation in online discussions in Piazza
- **Individual Homework Assignments:** These will be primarily programming and reflection assignments based on the concepts learned throughout the course

Assessment	Final Grade %

Weekly quizzes	30%
Project assignments	60%
Class participation	10%

Course and Grading Policies

- **Late-work policy**: All work is expected to be handed in at the indicated due date and time. For fairness to the whole class, no late submissions will be accepted for the weekly quizzes. The penalty for turning in project assignments late is 10%/day.
- **Participation policy**. Class participation will be graded by in-class engagement, including attendance and punctuality, asking relevant questions based on a critical review of required readings and lectures, preparation for any in-class exercises, and responses on the class discussion board. The lack of attendance and participation, will count against your participation grade.

Course Schedule. The following schedule provides a general overview of topics and assignments.

Week	Topics	Assignments	Notes
1	• Course introduction • Distributed systems and SOA • The Microservice architecture style • Microservice in practice • Microservice benefits and challenges • Microservice security	• Week 1 reading assignment • Week 1 quiz • A1 given (dockerized REST service on AWS)	
2	• Types of components and connectors in the Microservice style ○ REST vs component technology (gRPC, GraphQL, etc.) ○ REST API design • Asynchronous messaging (producers, consumers, queue/topics) • Core principles for microservice design: "IDEALS"	• Week 2 reading assignment • Week 2 quiz • A2 Given (BFF, SRP, JWT)	A1 due
3	• Service interceptors, API Gateway, BFF, service mesh • Availability strategies ○ Monitoring and exception tracking ○ Timeout and Retry ○ Circuit breaker ○ Bulkheads	• Week 3 reading assignment • Week 3 quiz • A3 Given (K8S, Kafka on AWS (MSK), circuit breaker)	A2 due
4	• Messaging patterns used in microservice architectures ○ Building blocks: point-to-point, pub-sub ○ For reliability: store-and-forward, transactional outbox, dead-letter channel ○ Other: sync-over-async, push vs pull • Event-Driven Architecture (EDA) and its tradeoffs ○ Choreography and orchestration • Asynchronous without a message broker • Post-SQL data stores	• Week 4 reading assignment • Week 4 quiz	

5	● Post-SQL data stores (cont.) ● Strategies for managing data ○ Service autonomy and distributed transactions ○ Saga pattern (compensation) ○ Database per Microservice pattern	● Week 5 reading assignment ● Week 5 quiz ● A4 Given (NoSQL (ES), data replication)	A3 due
6	● Strategies for managing data (cont.) ○ Data replication and eventual consistency ○ CAP theorem ● CQRS pattern ● Event sourcing ● Service loose coupling design strategies ○ Coupling ○ Wrapper pattern for services	● Week 6 reading assignment ● Week 6 quiz	
7	● Moving from a monolithic to a microservice architecture (Strangler Fig pattern) ● Scalability Defined	●	A4 due (Thursday before classtime)

References

[Bass13]	L. Bass, P. Clements, R. Kazman. *Software Architecture in Practice, Third Edition*. Addison-Wesley, 2013.
[Bianco07]	P. Bianco, R. Kotermanski, P. Merson. “Evaluating a Service-Oriented Architecture”. CMU/SEI-2007-TR-015, September 2007.
[Bianco11]	P. Bianco, G. Lewis, P. Merson, S. Simanta. ”Architecting Service-Oriented Systems”. CMU/SEI-2011-TN-008, August 2011.
[Clements10]	Clements P., F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond, Second Edition*. Addison-Wesley, 2010.
[Cockburn05]	Alistair Cockburn “Hexagonal Architecture.” 2005.
[Costa15]	Bruno Costa, Paulo Pires, Paulo Merson. “Evaluating REST Architectures - Approach, Tooling and Guidelines”. The Journal of Systems and Software. Elsevier, October 2015.
[docker18]	”Docker Overview”. Docker online documentation.
[Erl07]	T. Erl. *SOA Principles of Service Design*. Prentice Hall, July 2007.
[Erl09]	T. Erl. *SOA Design Patterns*. Prentice Hall, January 2009.
[Evans03]	E. Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional, 2003.
[Fielding00]	R. Fielding. *Architectural Styles and the Design of Network-Based Software Architectures*. PhD diss., University of California, Irvine, 2000.
[Foote97]	B. Foote and Joseph Yoder. "Big Ball of Mud". PLoP'97.
[Fowler04]	M. Fowler. "Strangler Fig Application". June 2004.
[Fowler10]	M. Fowler. "Richardson Maturity Model - steps toward the glory of REST". March 2010.
[Gamma94]	E. Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
[Helland18]	P. Helland. "Mind Your State for Your State of Mind". ACM Queue, volume 16 issue 3, July 2018.
[Hohpe03]	Hohpe, G. & Woolf, B. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003.
[Ibryam19]	Ibryam, B & Huß, R. *Kubernetes Patterns*. O’Reilly Media, 2019.
[Joshi21]	Unmesh Joshi, “Idempotent Receiver”, January 2021.
[Keeling17]	M. Keeling. *Design It!* Pragmatic Bookshelf, October 2017.
[Lewis14]	Lewis, J.; & Fowler, M. “Microservices”. 2014.

[Martin06]	R. C. Martin. *Agile Principles, Patterns, and Practices in C#.* Pearson, July 2006.
[Merson15a]	P. Merson. "Defining Microservices". SEI Blog, November 2015.
[Merson15b]	P. Merson. "Microservices Beyond the Hype: What You Gain and What You Lose". SEI Blog, November 2015.
[Merson16]	P. Merson. "Microservice design pattern". SOAPatterns.org, April 2016.
[Merson19]	P. Merson & J. Yoder. "Modeling Microservices with DDD". Presentation at SATURN 2019.
[Merson21]	P. Merson. "Principles for Microservice Design: Think IDEALS, Rather than SOLID". The InfoQ eMag Issue #91, February 2021.
[Meyer97]	B. Meyer. *Object-Oriented Software Construction, 2nd Edition*. Prentice Hall, 1997.
[Millett15]	S. Millett & N. Tune. *Patterns, Principles, and Practices of Domain-Driven Design*. Wrox, 2015.
[Millsap10]	C. Millsap. "Thinking Clearly About Performance, Part 2." Communications of the ACM, 53, 10 (October 2010): 39–45.
[Moussa19]	H. Moussa. "Business process visibility & event-driven services". December 2019..
[Newman15]	S. Newman. "Pattern: Backends For Frontends". November 2015.
[Newman19]	S. Newman, *Monolith to Microservices*. O'Reilly Media, November 2019.
[Newman21]	Newman, S. *Building Microservices Second Edition*. O'Reilly Media, August 2021.
[Nygard18]	M. Nygard. *Release It!, 2nd Edition*. O'Reilly Media, January 2018.
[OWASP21]	Top 10 Web Application Security Risks by owasp.org
[Parnas72]	D. Parnas. "On the criteria to be used in decomposing systems into modules". Communications of the ACM, vol. 15, no. 12, December 1972, pp. 1053-1058.
[Richardson16]	Richardson, C. "Database per Microservice pattern".
[Richardson18]	Richardson, C. *Microservice Patterns*. Manning, 2018. (See also: microservices.io.)
[Roberts18]	M. Roberts. "Serverless Architectures". May 2018.
[Rotem-Gal-Oz12]	Rotem-Gal-Oz, A. *SOA Patterns*. Manning, 2012.
[Sahni15]	Vinay Sahni. "Best Practices for Designing a Pragmatic RESTful API".
[SOA09]	"SOA Manifesto". 2009.
[Vernon13]	V. Vernon. *Implementing Domain-Driven Design*. Addison-Wesley Professional, February 2013.
[Vernon21]	V. Vernon and T. Jaskula. Strategic Monoliths and Microservices: Driving Innovation Using Purposeful Architecture. Addison-Wesley Professional, October 2021.

[Yoder19]	J. Yoder, A. Aguiar, P. Merson, H. Washizaki. “Deployment Patterns for Confidence”. Asian PLoP. March 2019.
[Yoder20]	J. Yoder & P. Merson. “Strangler Patterns”. PLoP. October 2020.

Accommodations for Students Disabilities. If you have a disability and have an accommodations letter from the Disability Resources office, I encourage you to discuss your accommodations and needs with me as early in the semester as possible. I will work with you to ensure that accommodations are provided as appropriate. If you suspect that you may have a disability and would benefit from accommodations but are not yet registered with the Office of Disability Resources, I encourage you to contact them at access@andrew.cmu.edu.

Academic Integrity. Honesty and transparency are important to good scholarship. Plagiarism and cheating, however, are serious academic offenses with serious consequences. If you are discovered engaging in either behavior in this course, you will earn a failing grade on the assignment in question, and further disciplinary action may be taken.

For a clear description of what counts as plagiarism, cheating, and/or the use of unauthorized sources, please see the University’s Policy on Academic Integrity.

If you have any questions regarding plagiarism or cheating, please ask me as soon as possible to avoid any misunderstandings. For more information about Carnegie Mellon’s standards with respect to academic integrity, you can also check out the Office of Community Standards & Integrity website.

Student Wellness. As a student, you may experience a range of challenges that can interfere with learning, such as strained relationships, increased anxiety, substance use, feeling down, difficulty concentrating and/or lack of motivation. These mental health concerns or stressful events may diminish your academic performance and/or reduce your ability to participate in daily activities. CMU services are available, and treatment does work. You can learn more about confidential mental health services available on campus at the Counseling and Psychological Services website. Support is always available (24/7) from Counseling and Psychological Services: 412-268-2922.

We must treat every individual with respect. We are diverse in many ways, and this diversity is fundamental to building and maintaining an equitable and inclusive campus community. Diversity can refer to multiple ways that we identify ourselves, including but not limited to race, color, national origin, language, sex, disability, age, sexual orientation, gender identity, religion, creed, ancestry, belief, veteran status, or genetic information. Each of these diverse identities, along with many others not mentioned here, shape the perspectives our students, faculty, and staff bring to our campus. We, at CMU, will work to promote diversity, equity and inclusion not only because diversity fuels excellence and innovation, but because we want to pursue justice. We acknowledge our imperfections while we also fully commit to the work, inside and outside

of our classrooms, of building and sustaining a campus community that increasingly embraces these core values.
Each of us is responsible for creating a safer, more inclusive environment.
Unfortunately, incidents of bias or discrimination do occur, whether intentional or unintentional. They contribute to creating an unwelcoming environment for individuals and groups at the university. Therefore, the university encourages anyone who experiences or observes unfair or hostile treatment on the basis of identity to speak out for justice and support, within the moment of the incident or after the incident has passed. Anyone can share these experiences using the following resources:

- **Center for Student Diversity and Inclusion:** csdi@andrew.cmu.edu, (412) 268-2150
- **Report-It online anonymous reporting platform:** reportit.net username**:** *tartans* password**:** *plaid*

All reports will be documented and deliberated to determine if there should be any following actions. Regardless of incident type, the university will use all shared experiences to transform our campus climate to be more equitable and just.