Advanced Distributed Systems (DS2)

-- Research Seminar --

https://columbia.github.io/ds2-class/

Interested in...

1. Big data?
2. Cloud computing?
2. Scalable web services?
4. Plus the large-scale infrastructure systems that are making these possible?

If so, you should be interested in distributed systems!

Distributed Systems Are Everywhere!

- Most of today's services and apps are powered using distributed systems:
  - Systems that span the world,
  - Serve millions of users,
  - Store & process petabytes or exabytes of data,
  - And are always up!
- But also pose some of the hardest CS problems today.
  - Companies address the challenges by combining decades' worth of research in distributed systems with painstaking engineering.

What Is a Distributed System (DS)?

- Multiple interconnected computers (aka, nodes) that coordinate to provide a common, coherent service.

Why DSes?

1. To increase capacity
2. To get reliable, always-on systems
3. To improve performance

- Achieving these properties is *hard*. Easy to get the opposite trends from distribution ☺.
- Decades of DS research have resulted in key protocols, programming abstractions, system architectures, and semantics to address the challenges.
- This class reviews some of the foundational works.

1. DS to Increase Capacity

- No single machine can handle the data and request loads that most services face today.
- Idea: Shard the data/requests on multiple machines.
  - Together the machines should be able to handle more data/ requests than individually.
  - This is called scalability.
- Challenge: Achieving scalability is hard:
  - the application may not exhibit sufficient parallelism
  - bottlenecks may inhibit parallelism
  - nodes need to coordinate, and that's expensive.
- This class: Scalable programming models, architectures, and coordination protocols.

2. DS to Increase Reliability

- At scale, failures are inevitable and continuous.
- Idea: Replicate data/requests on multiple machines.
  - Service should continue despite failure of some of the replicas.
  - This is called fault tolerance.
- Challenge: Achieving fault tolerance is hard:
  - many types of failures (network vs. machine, delayed vs. crashed, detectable failure vs. silent corruption, …)
  - different failures may demand different actions, but often one cannot tell them apart.
- This class: Replication protocols that are guaranteed to work under broad classes of failures.

Typical first year for a cluster (Jeff Dean, Google, 2008)

~1000 individual machine failures

~1000s of hard drive failures - slow disks, bad memory, misconfigured machines, flaky machines, etc.

~0.5 overheating events (power down most machines in <5 mins, ~1-2 days to recover)

~1 power unit failure (~500-1000 machines suddenly disappear, ~6 hours to come back)

~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)

~1 network rewiring (rolling ~5% of machines down over 2-day span

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~5 racks go wonky (40-80 machines see 50% packetloss)

~8 network maintenances (4 might cause ~30-minute random connec)vity losses)

~12 router reloads (takes out DNS and external vips for a couple minutes)

~3 router failures (have to immediately pull traffic for an hour)

~dozens of minor 30-second blips for dns

Leslie Lamport (c. 1990):

“A distributed system is one where the failure of a computer you didn’t know existed renders your own computer useless”

3. DS to Increase Performance

- Performance matters to users, and delaying service for even a few tens of milliseconds can lose customers.
- Idea: Cache or replicate data/service close to clients.
- Challenge: Often scalability and replication go against performance/efficiency.
- This class: Distributed performance evaluation and bottleneck finding.

Overarching Challenge: Semantics

- Sharding, replication, and caching all raise substantial consistency/semantics challenges in the context of failures.
- A key goal of distributed systems is to provide clear and meaningful semantics.
  - The golden standard is usually a centralized system's semantic.
  - But there is usually a spectrum of semantics, from weaker to stronger.
  - This spectrum usually creates programmability vs. performance tradeoffs.
- This class: understand the spectrum of semantics for certain types of services, the performance and programmability tradeoffs they introduce, and key mechanisms/protocols for how to achieve them.

DS Classes at Columbia

1. Fundamentals of Distributed Systems (RG, Fall)
   - 4000-level class
   - teaches basic concepts, principles of large-scale, distributed-system design
   - discusses application of those concepts/principles in real-world systems (e.g., Google/Facebook/Amazon)
2. Advanced Distributed Systems (RG, Spring)
   - 6000-level class, i.e. PhD-level research seminar
   - discusses these topics in depth by reading the most influential papers that originated them
   - includes both classical papers on well-understood topics and new, state-of-the-art papers

This class (Advanced DS)

- Exclusively focused this year on paper reading, reviewing, and discussion (100% of the grade!).
- One topic per class, two representative papers.
- Students must provide written responses to specific questions plus a novel contribution for which they will argue in class.
- In class, students will play assigned roles in the discussion, but can contribute more broadly.
- The class organization and assigned roles are taken from Ethan Katz-Bassett's Internet Delivery Seminar (ELEN6775) and tweaked for the distributed systems focus of this class.

Paper Reviews, Comments, Discussions

- For each paper:
  - Review (due 12pm Thursdays): HotCRP form of ~10 questions, each answer a few sentences.
  - Novel response (due 8pm Thursdays): Leave a HotCRP comment summarizing the contribution you will make in class as part of your assigned role.
  - In class (Fridays), you must participate in discussion in your assigned role, but you can also contribute any other ideas/questions you might have.
- For all three, grading will be on 0-2 scale:
  - 0 = absent, late, or unreasonably poor-quality review/comment/discussion.
  - 1 = reasonable but not concise or particularly insightful or comprehensive.
  - 2 = high quality and insightful review/comment/discussion.
- You can skip two papers with no points deducted.

Reviews (due 12pm Thursdays)

1. What problem is the paper solving?
2. Why is that problem important?
3. What was the previous state of the art?
4. How does this paper advance the state of the art?
5. How does the system work?
6. What are the key insights in the design?
7. How is it evaluated?
8. What are the key results?
9. What related problems are still open?
10. [optional] What other comments do you have?
11. [only seen by RG] Should this paper be on syllabus?

Novel Response (due 8pm Thursdays)

- In addition to answering review questions, leave a brief statement of the points you want to contribute to the discussion, based on the role(s) assigned to you.
- Leave a HotCRP comment with your novel response.
  - A few paragraphs, less than a page.
- Assigned one or two of ~12 roles for each paper:
  – We rotate roles, so you won't have the same role for all papers.
  – The number of roles each student plays for each paper depends on the number of students in the class.
  – Write your role at the top of the novel response.

Discussion Roles

Roles 1-5: *-Synthesizer

1. Problem-synthesizer (review questions 1&2)
   - What problem is it solving?
   - Why is that important?
2. State-of-the-art-synthesizer (review questions 3&4)
   - What was the previous state-of-the-art?
   - How, and how much does this system advance state of art?
3. How-synthesizer (review questions 5&6)
   - How does the system work?
   - What are the key insights?
4. Results-synthesizer (review questions 7&8)
   - How was the technique evaluated? What were key results?
5. Open synthesizer (review question 9)
   - What related problems are open/is this problem fully solved?

*-Synthesizer Instructions

- Read over everyone else's review responses for your category (including your own).
  - Synthesize the consensus response to those questions.
  - Identify outlier/contrasting opinions (list name of student who had the opinion: "In contrast, X thought that...").
  - Write down these points in the position comment.

- In class:
  - Present the consensus response to questions.
  - Invite discussion, especially of outlier/contrasting opinions (call on students by name).

Role 6: Revealer

- If you could get the authors to reveal an additional aspect(s) of their design:
  - Which aspect would it be to help us best understand the system?
  - Which aspect would it be to help us design a followup/ improvement/expansion?
  - (two separate questions)
- For each:
  - Why did you pick that aspect, and why will it be revealing?
  - What are the range of possibilities for what it might show, and what would the implications be for different results? (note that, if it is clear what will be revealed, it is probably not the most important thing to reveal)

Role 7: Redesigner

- Propose modifications to the proposed design and/or to its assumptions/use cases/workloads that you believe would make the design better from *some* perspective.
  - Describe what changes you are proposing.
  - Describe how you expect the modifications to improve the system and why.
  - Identify any tradeoffs the modifications might introduce.
  - Identify assumptions under which you expect your redesign to be better than theirs and vice versa.
  - Identify situations/application domains under which the assumptions that favor your design are plausible.

- Must come up with a rough evaluation plan to evaluate the hypothesized improvement/tradeoffs.

Role 8: Follower-On

- What would your follow-on research to this paper be? Ex:
  - Improve an algorithm in the paper; or
  - Find a new, significant evaluation question for it; or
  - A brand-new way to address the same problem/goal.
- Must propose a brand-new design (worth another paper).
  - This is different from the redesigner, who only needs to propose an improvement for this paper.
- It's a good idea to always approach reading a paper from this perspective.

Role 9: Retrofitter

- Find a different, interesting application/problem for the proposed system or some of its mechanisms.
  - Describe why the new application is interesting, why we should care about it.
  - Describe your new design plus a rough evaluation plan for it.
  - Describe how the design will need to change to apply it in that setting. Propose a specific redesign for that new purpose.
  - Describe why you believe the your design (inspired by the authors' own design) is a good fit for the new problem you're using it for.
- Must propose new system design & rough evaluation plan (see later).

Role 10: Surveyer

- Briefly describe a few other works on the same topic.
- We include some “optional resources” for each topic, you must summarize and connect those to this paper.
- In addition, find at least one more paper on your own and summarize/connect it to this paper.
  - You can search for recent related papers (that come after this work).
  - You can describe the context of the paper (please don’t just summarize the related work section of the paper in question, that’s the role of the state-of-the-art synthesizer; you need to find another paper and read it).
- Give complete citations and links for each paper you cover at the end of your comment.

Role 11: Connector

- Connect this paper to a previous paper we have read this semester.
  - You could compare/contrast
  - You could use an idea from an earlier paper here
  - You could use an idea from this paper in an earlier paper
  - You could discuss how to merge two approaches to yield greater benefit
  - You could try porting this paper to the earlier papers setting, or vice versa

Role 12: Hater

- What is the weakest part of the paper? Give one example of each of two types of weaknesses:
  - Weakest part of the technical work, for example a limitation of the approach or a questionable assumption.
  - Weakest part of the paper's presentation of the work, for example a portion that is not explained well.
  - Try to avoid listing a missing evaluation, as we have a separate role for that.
- Why is it weak? Does that undermine the whole paper?
- Can you find something in the paper that actually BREAKS?

Unassigned Role: Wildcard

- Any other interesting and insightful idea you might have based on the paper. Preferably not fitting into any of the other categories.

- This role is not assigned, but you can grab it *in addition to your assigned role(s) for a paper*.
  - If your idea/contribution is worth a 2 score (interesting, insightful, opens up a lot of conversation, etc.), then it acts as extra credit. I will not award a 1 score to wildcard responses.
  - For example, you can skip the novel response for another paper, or you can make up for a 1-level response you gave for a different paper.
  - If you adopt the wildcard role, you must register the response as a separate comment in HotCRP, and you must invoke your wildcard role explicitly in class.

In-Class Discussions

- Assume everyone has read each paper.
- Everyone is expected to share their novel responses.
  - Share your thoughts without just reading what you wrote.
  - Summarizers call on others who made interesting points.
- Reply to interesting ideas, ideas you disagree with, ideas you don't understand, etc.
  - Goes into your 0/1/2 score for class discussions.
- If useful, feel free to prepare slides.
  - Use Google Slides, a single slideshow per paper.
  - Share in HotCRP comment, editable by anyone at Columbia.
  - Feel free to borrow images from paper or from author's slides, just provide a citation.

Option: A Project for 50% of the Grade

- Expect the class to be very heavy in terms of reading load: every week, you'll have two papers (sometimes more, depending on role) to read, review, and come up with novel responses, which you then defend in class.
- If you'd prefer to replace 50% the reading load with "doing" load, you can elect to do a research project.
- The catch:
  - You will not get to choose the topic, you will have to pick one from my list.
  - You will need to do it REALLY WELL!
  - You must commit to a project by **February 8**, so I can reassign half of your roles to others (you can choose dates when you want your roles reassigned, but you have to do it a priori).
  - You can still submit your reviews, assume the wildcard role, and participate in class for all papers. Any 2-score contributions will add up as extra credit.

Need TA

- Assign roles to students in rotation. Reassign roles as the class size fluctuates and when some students elect to do projects.
- Mark roles, configure deadlines, etc. on HotCRP.
- Grade reviews (not his/her own).
- I will grade novel responses and in-class discussions.

- Please volunteer!

URLs & Role IDs

- Class website:
  - https://columbia.github.io/ds2-class/
- HotCRP website:
  - https://cucs-ds2-2019.hotcrp.com/
- Mapping of roles to IDs that I will use on HotCRP:

1=problem-synthesizer
2=state-of-the-art-synthesizer
3=how-synthesizer
4=results-synthesizer
5=open-synthesizer
6=revealer
7=redesigner
8=follower-on
9=retrofitter
10=surveyer
11=connector
12=hater

Acknowledgements

- The reviewing structure is adapted from Ethan Katz-Bassett's course on Internet Service Delivery.

- Statistics I quoted in the slides or in class are taken from:
  - https://www.facebook.com/notes/facebook-engineering/scaling-facebook-to-500-million-users-and-beyond/409881258919/
  - https://en.wikipedia.org/wiki/Exabyte#Google