### ***CS 736: Tuesday, February 13, 2018 – Crash Consistency***

**All File Systems Are Not Created Equal:   
On the Complexity of Crafting Crash-Consistent Applications**  
[Thanumalayan Sankaranarayana Pillai](http://www.cs.wisc.edu/~madthanu/), [Vijay Chidambaram](http://www.cs.wisc.edu/~vijayc/), [Ramnatthan Alagappan](http://www.cs.wisc.edu/~ra/),   
[Samer Al-Kiswany](http://www.cs.wisc.edu/~samera/), [Andrea C. Arpaci-Dusseau](http://www.cs.wisc.edu/~dusseau/), [Remzi H. Arpaci-Dusseau](http://www.cs.wisc.edu/~remzi/)  
*11th Symposium on Operating Systems Design and Implementation (OSDI '14)*

1. Imagine you have some initial state (/x/f1 contains “a foo”) and your application wants to pwrite(/x/f1, 2 (offset), “bar”) so that the only two possible final states are “a foo” or “a bar” even if a crash occurs. Assume each character represents one block. What operations must your application do to be robust on all POSIX-compliant file systems?
2. What is the drawback of creating applications that work correctly on all file systems?
3. What are persistence properties? How can they be determined?   
   Which file systems seem to provide the most guarantees?
4. What assumptions does the Git add protocol make about ordering and atomicity of the underlying file system? Which assumptions are the least often true?
5. How many vulnerabilities does Git exhibit on different file systems?

**Optimistic Crash Consistency**  
[Vijay Chidambaram](http://www.cs.wisc.edu/~vijayc/), [Thanumalayan Sankaranarayana Pillai](http://www.cs.wisc.edu/~madthanu),   
[Andrea C. Arpaci-Dusseau](http://www.cs.wisc.edu/~dusseau/), [Remzi H. Arpaci-Dusseau](http://www.cs.wisc.edu/~remzi/)   
*Proceedings of the 24th ACM Symposium on Operating Systems Principles (SOSP '13)*

1. What is the philosophy of pessimistic journaling? What are its problems?
2. What is probabilistic crash consistency? Why is it used?   
   When does it have the most problems?
3. What are the goals of optimistic crash consistency? What is the idea behind it?
4. What is the idea behind Asynchronous Durability Notifications? What is their problem?
5. Figure 6: Can you explain why:
   1. M:1 cannot be written? What could happen if it were written?
   2. Assume receive ADN for D:1. M:2 cannot be written? What could happen?
   3. Assume receive ADN for D:1. M:3 cannot be written? What could happen?
6. How are checksums used to remove a flush from the pessimistic protocols?
7. Why must optimistic journaling wait for an ADN before writing M?   
   Why might this waiting not have a significant impact on performance?
8. ~~Details: Why could overwriting a data block with new file contents lead to problems? How is this problem avoided in OptFS?~~
9. ~~Details: In any journaling system, what is good and bad about full data journaling? How could an append be handled with reuse after notification? How is it actually handled in OptFS?~~
10. What does fsync(file) usually do? What does fsync() conflate? What does OptFS introduce?
11. Are there any workloads where OptFS performs worse than ext4 with flushes?
12. Are there any workloads where OptFS leads to inconsistencies?