### ***CS 736: Thursday, February 8, 2018 – SSD-Conscious Storage***

### *Da Zheng and Disa Mhembere and Randal Burns and Joshua Vogelstein and Carey E. Priebe and Alexander S. Szalay*, [FlashGraph: Processing Billion-Node Graphs on an Array of Commodity SSDs](https://www.usenix.org/system/files/conference/fast15/fast15-paper-zheng.pdf) Conference on File and Storage Technologies (FAST 2015)

1. What does it mean for a graph-processing engine to be semi-external?
2. How does FlashGraph describe previous work of GraphChi and X-stream?   
   Do you agree with their descriptions?
3. What are the four goals of the FlashGraph design?   
   Do they show that they meet these goals?
4. Do you find anything interesting about the programming model of FlashGraph (compared to previous systems)? Does it contain any optimizations or features that seem useful?
5. What data structures does FlashGraph keep in memory (instead of SSD)?   
   How does FlashGraph minimize its memory usage?
6. How does FlashGraph minimize accesses to SSD?
7. To optimize performance, in what order should vertices be scheduled? Why?
8. In the performance evaluation, is FlashGraph compared to reasonable configurations of GraphChi and X-stream? Is FlashGraph always better?
9. Are there any experiments you wish they would have shown?
10. At a high level, how are the techniques used by FlashGraph and WiscKey similar?

### [*Lanyue Lu*](http://www.cs.wisc.edu/~ll/)*,*[*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/) [WiscKey: Separating Keys from Values in SSD-conscious Storage](http://research.cs.wisc.edu/adsl/Publications/wisckey-fast16.pdf) *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST '16)*

1. Why do LSM-trees perform well on HDDs? What different characteristics about SSDs encourage a change in LSM design?
2. Why do LSMs incur write amplification? Why do they incur read amplification?
3. How can one achieve nearly sequential bandwidth out of an SSD for random operations?
4. Why might one think that separating keys from values would hurt performance?   
   Why does separating keys and values improve performance?
5. Why is it challenging to handle range queries in WiscKey? How are they handled efficiently?
6. How does WiscKey determine a value is garbage and should be reclaimed?   
   What is similar about this compared to LFS?
7. How is the performance of small put() operations improved?
8. Does WiscKey performance relative to LevelDB increase or decrease as the value size increases? Why?
9. As shown in the experiments, what hurts/helps the performance of range queries in WiscKey? Do you have any ideas for how to help the difficult situation?
10. Are there any experiments you wish they would have shown?