A. Arpaci-Dusseau **Department of Computer Science**
CS739: Distributed Systems **University of Wisconsin, Madison**

Wide-area cooperative storage with CFS - SOSP'01

1 Introduction

Very similar to paper "Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility" by Antony Rowstron (Microsoft Research), Peter Druschel (Rice University), SOSP'01

- How would you characterize a *peer-to-peer (P2P)* system?
- What were the goals of CFS?
- What applications did they target?

2 Design Overview

- What is the overall architecture of their system? (i.e., what are the three layers and what are their responsibilities?) Which layers were implemented? Is there a clean separation between layers?
- Figure 2: How is a CFS file system structured? How does a publisher insert blocks into CFS? What happens if a publisher wants to update blocks? How and why is the *root block* treated differently?
- Figuring out the correct way to deal with *delete* operations can be tricky in distributed systems. Why is having an explicit delete non-trivial in distributed systems? How have other systems (e.g., Google FS) dealt with the possible problem? In contrast, CFS chose to, in effect, "lease" storage space; the publisher can repeatedly ask for extensions. What are the advantages of this approach? Disadvantages?

3 Chord Layer

- CFS uses Chord to locate blocks on a specific node based on the content-hash of the data to a key. The Chord lookup layer, overlay network, was introduced in SIGCOMM'01 paper. At a high-level, how does Chord know which node is responsible for a given key? What attractive properties does Chord provide?
- In more detail: What is the role of the successor list? How does a new node join a Chord ring? What is the point of the finger table?
- What is *server selection* and why is it needed?

- P2P systems need to worry about malicious nodes. Can a malicious node alter data contents? What can a malicious node do? Why can't you let nodes pick their id?
- Why does CFS use virtual nodes? What is the danger of allowing virtual nodes?

4 DHash Layer

- CFS stores files in *fixed-sized blocks*. In contrast, Pastry stores entire files on the same storage nodes. What are the pros and cons of using fixed-sized blocks?
- Since nodes are free to leave the system at any time, a P2P system must take special care to ensure *availability*. How does CFS do this?
- *Caching* is used to improve performance in CFS. How is it is performed? Do you think this will work well?

5 Implementation

- Anything strike you as interesting in the implementation?

6 Experimental Results

- What parts of CFS are they (not) testing?
- For their "Real life" experiments, how many machines are they running on? What does the workload look like? What are they showing in Figure 6? For server selection, what are they selecting between? What is the point of Figure 7? What is the point of figure 8?
- What do they mean by "Controlled Experiments"? Point of Figure 9? Point of Figure 10? Point of Figure 11? Point of Figure 12?
- Next set of experiments use controlled setting to investigate failures. Workload inserts 1000 blocks on 1000 servers with 6 replicas, they then fail some percentage of servers, and then initiate 1000 fetches while Chord is rebuilding. What does Figure 13 show? What does Figure 14 show? What doesn't Figure 14 include in the average RPC count? (Does this seem right?) Figure 15 includes this data, but averaged over 1000 requests...

7 Conclusions

- Very preliminary work, but started an avalanche of related work. Many follow-on papers look at variations/optimizations and sensitivity to churn for different routing algorithms. Are the lessons learned from building P2P systems applicable to building large-scale distributed systems in more controlled environments?