Multi-structured redundancy

Eno Thereska, Phil Gosset, Richard Harper
Microsoft Research, Cambridge UK
[HotStorage'12]

Motivation

Language/APIs

{put, get,...}, …, {create, read, write, …}, …, {addNode, AddEdge, GetNeighbors, …}, …, SQL

Transactions, consistency, atomicity

No/transactions, eventual/strong consistency, rename, ...

Caching and prefetching

No caching, no prefetching, LRU, MRU, hint-guided prefetching, ...

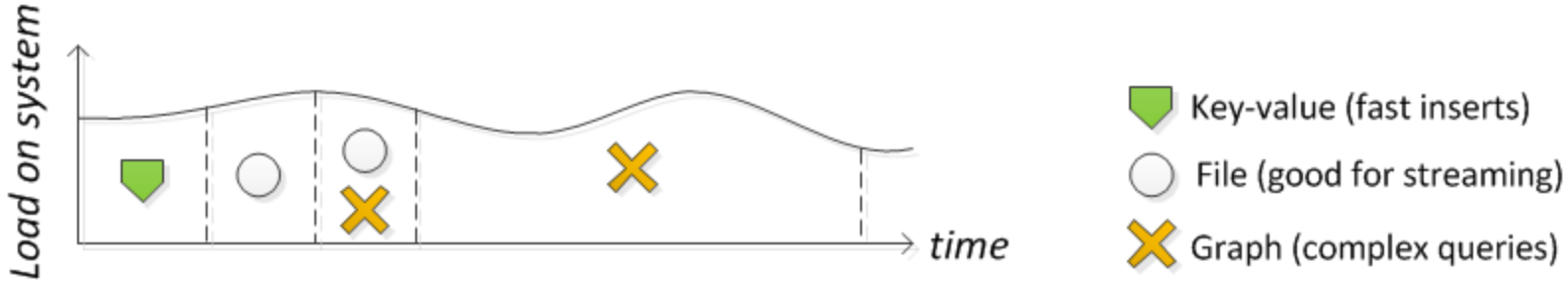
Layout on persistent store or in memory

log-structured, Btree, co-located, row store, column store, matrix, ...

Over-optimizing point solutions

- We've done it in the past with file systems (FS)
- We're doing it now with key-value (KV) stores
- Maybe we'll do it with graph stores (GS)
- Do our assumptions stay true over time?
  - What if "value" in KV grows over time? (-> FS)
  - What if relations are desired among items? (-> GS)
  - What if our workloads change?
    - Facebook's photo store: [designed for scenarios where] "data is written once, read often, [and is] never modified" [Haystack, OSDI'10]

Workloads are too complex for a single abstraction



- Anecdotal evidence:
  - Home user uploads 100 photos, and later sorts, tags, and browses them
  - A small business processes transactions during the day and data mines them during the night

Real evidence

- Found mostly in database community
  - FILESTREAM addition to SQL Server [>2008]
  - Fractured mirrors (column-stores and row-stores) [VLDB'02]
- Other related work
  - SwissBox [CIDR'11]
  - TableFS [CMU'12]
  - Anvil [SOSP'09], Stasis [OSDI'06], BoxWood [OSDI'04]
  - WinFS attempt

Research agenda

- Investigate how multiple data structure abstractions can co-exist in the same system
- Two simultaneous paths
  - Data-center scale
  - Single-laptop/tablet scale

Outline

- Motivation
- Data-center store
- Laptop/tablet store

Data center approach

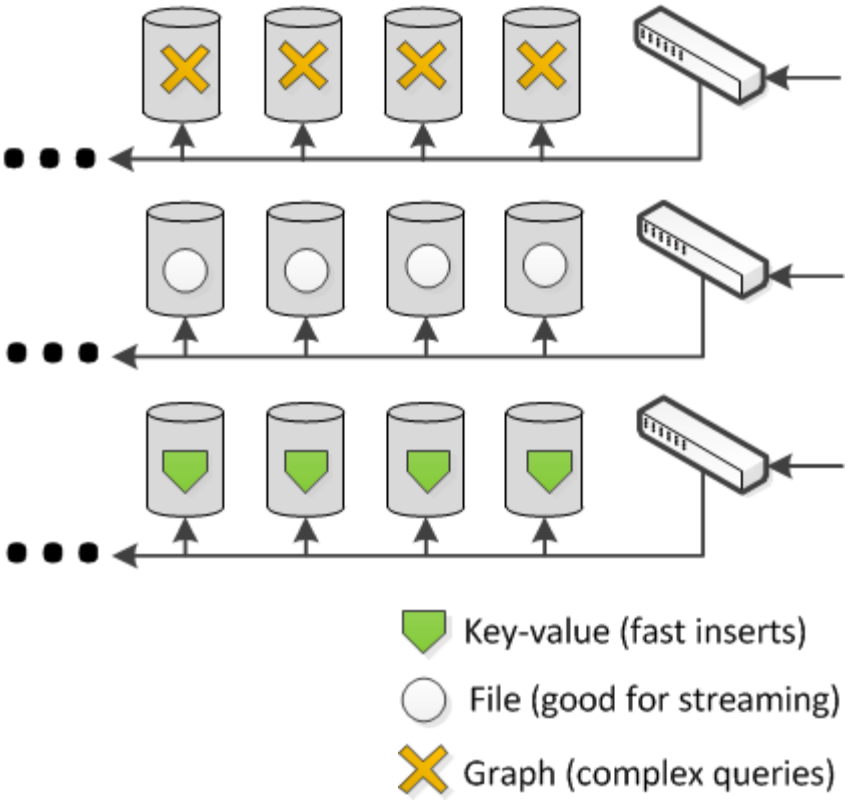
- Key idea: use existing N way redundancy
  - To have N data structures co-exist
- Analogies in PL
  - Sorted list and hash table storing the same data
  - N-way programming
- Investigating 3 data structures
  - Key-value, file and graph

No change in programming APIs (yet)

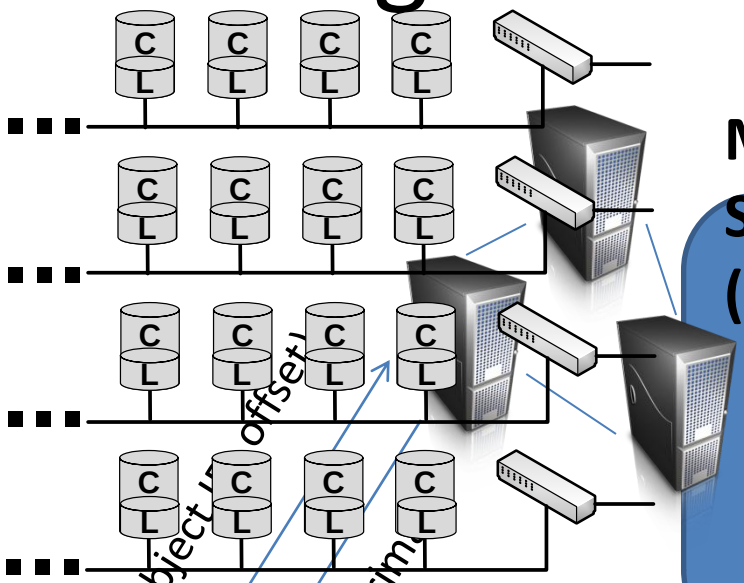
Data structure	APIs
Key-value	*put(), get(), delete()*
File	*create(), read(), write(), delete()*
Graph	*addNode(), addEdge()*
	getNeighbors(), delete([node/edge])

Table 1: The API into CamFS. Internally these calls are mapped to appropriate caching, prefetching and data lay-out building blocks.

Example


Key-value (fast inserts)
File (good for streaming)
Graph (complex queries)

Building on Everest[OSDI'08], Sierra[Eurosys'11]



Metadata Service (MDS)

Namespace, data location
Highly available (replicated)
Scalable & lightweight
Not on data path
Transactions

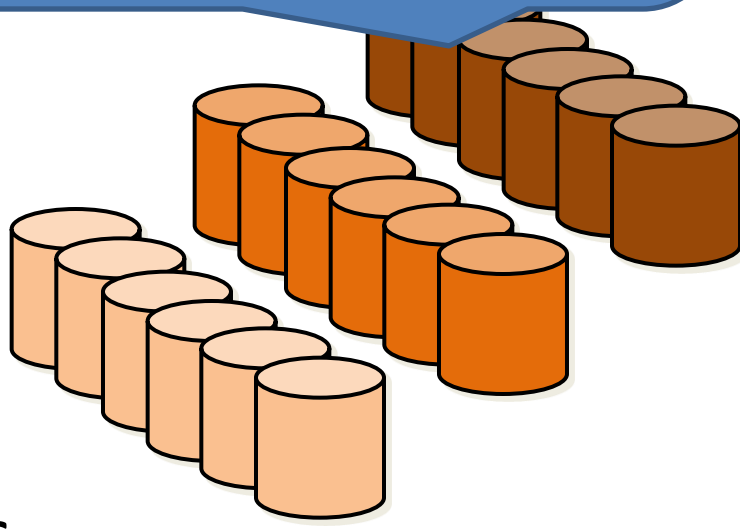
lookup(object "…", offset)

chunk ID, prim…

read(chunk ID, offset, size…)
write(chunk ID, offset, size, data...)

KV, File, Graph API

Client library



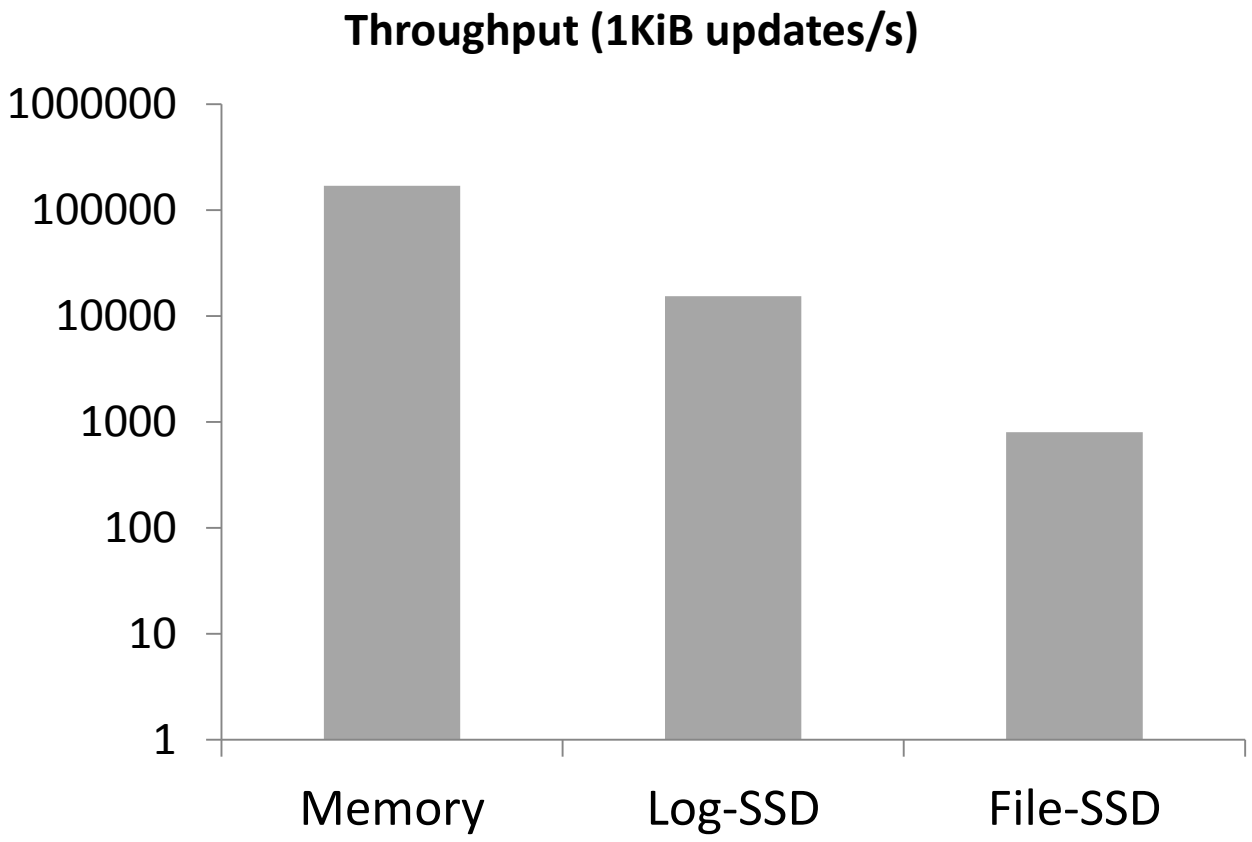
Servers

Challenges

- Are N data structures sufficient?
- Could peak performance suffer?
- What about performance interference?
  - SSDs and (NV)RAM only

Challenges (cont…)

- Speed-matching of updates



Challenges (cont…)

- Can recovery be uniformly fast?
- Could one single (in-memory) structure always be best? (e.g., RamCloud [SOSP'11] approach)

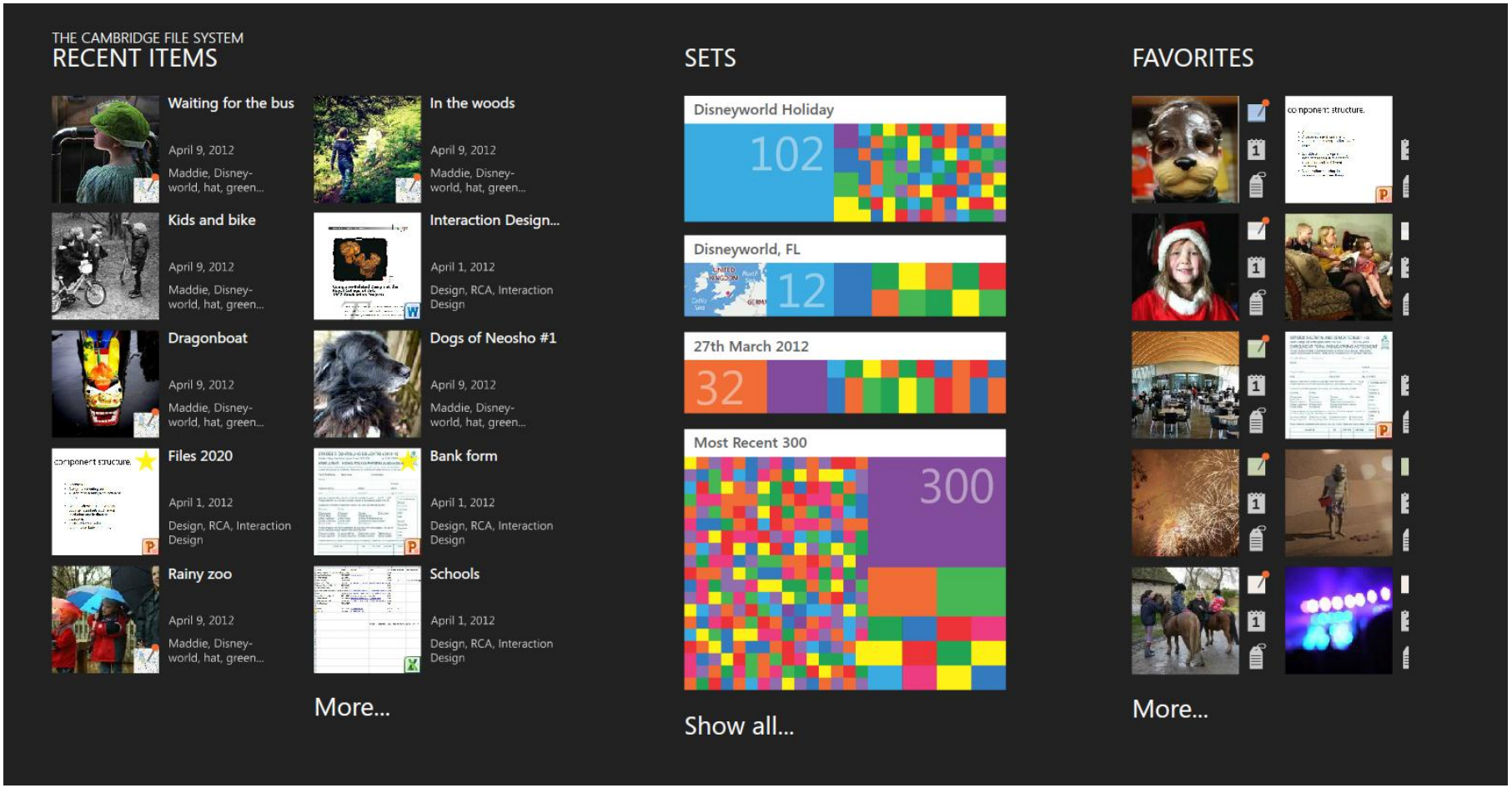
Outline

- Motivation
- Data-center store
- Laptop/tablet store

Laptop/tablet approach

- Why care?
  - People's stuff is not just files [CHI'12]
  - E.g., desire to natively store region of Facebook graph "locally"
  - Blur the lines between local file system and cloud
- Store exports native data structures
  - Key-value store, files, graph
  - No redundancy available, system uses partitioning

What we're building



Summary of research agenda

- Investigate how multiple data structure abstractions can co-exist in the same system
- Two simultaneous paths
  - Data-center scale
  - Single-laptop/tablet scale

Collaboration between Systems and Networking and Socio-digital Systems group
http://research.microsoft.com/sysdes/