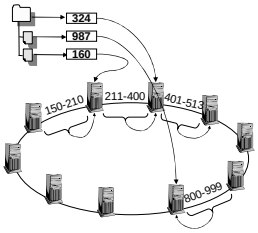
Defragmenting DHT-based Distributed File Systems

Jeffrey Pang, Srinivasan Seshan
Carnegie Mellon University
Phillip B. Gibbons, Michael Kaminsky
Intel Research Pittsburgh
Haifeng Yu
National University of Singapore

1

Traditional DHTs

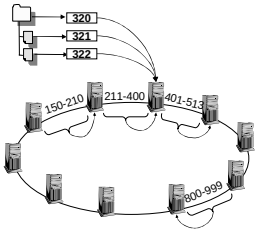
- Each node responsible for random key range
- Objects are assigned random keys



2

Defragmented DHT (D2)

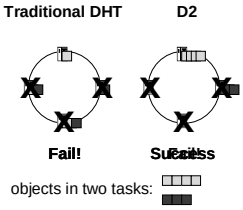
- Related objects are assigned contiguous keys



3

Why Defragment DHTs?

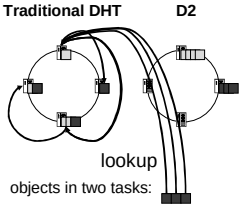
- Improves *Task Locality*
- Task Success Rate
  - Depend on fewer servers when accessing files
- Task Performance
  - Fewer DHT lookups when accessing files



4

Why Defragment DHTs?

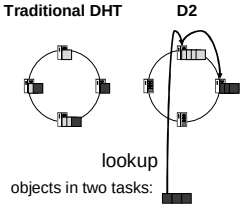
- Improves *Task Locality*
- Task Success Rate
  - Depend on fewer servers when accessing files
- Task Performance
  - Fewer DHT lookups when accessing files



5

Why Defragment DHTs?

- Improves *Task Locality*
- Task Success Rate
  - Depend on fewer servers when accessing files
- Task Performance
  - Fewer DHT lookups when accessing files



6

D2 Contributions

- Simple, effective locality techniques
- Real defragmented DHT implementation
- Evaluation with real file system workloads
- Answers to three principle questions…

7

Questions

- Can task locality be maintained simply?
- Does locality outweigh parallelism?
- Can load balance be maintained cheaply?

8

Questions

- Can task locality be maintained simply?
- Does locality outweigh parallelism?
- Can load balance be maintained cheaply?

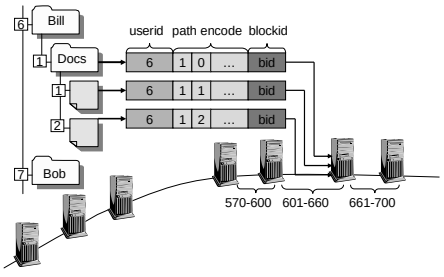
9

Technique #1: Locality Preserving Keys

- Preserve locality in DHT placement
  - Assign related objects nearby keys
- Group related objects
  - Leverage *namespace locality*
  - Preserve in-order traversal of file system
  - E.G., files in same directory are related

10

Assigning Object Keys



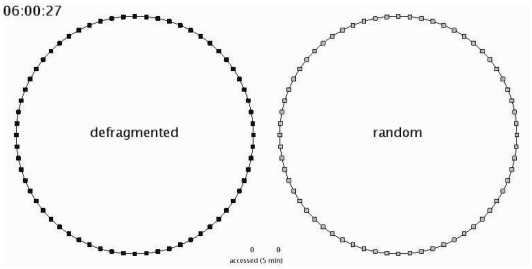
11

Practical Concern

- Key distribution is not uniformly random
  - ⇒ Object placement is no longer uniform
  - ⇒ Load imbalance if node IDs are random
- Solution:
  - Dynamically balance load (Discussed later in talk)

12

DHT Nodes Accessed



13

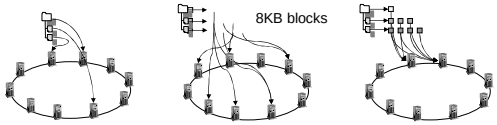
Task Availability Evaluation

- How much does D2 reduce task failures?
- Task = sequence of related accesses
  - E.G., '`ls -l`' accesses directory and files' metadata
  - Estimate tasks:

- Evaluated infrastructure DHT scenario:
  - **Faultload**: PlanetLab failure trace (247 nodes)
  - **Workload**: Harvard NFS trace

14

Systems Compared



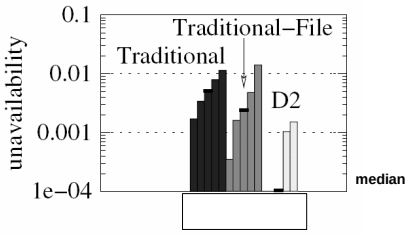
Traditional-File (PAST)

Traditional (CFS)

D2

15

Task Unavailability



What fraction of tasks fail?

16

Questions

- Can task locality be maintained simply?
- Does locality outweigh parallelism?
- Can load balance be maintained cheaply?

17

Technique #2: Lookup Cache

- Task locality ⇒ objects on a few nodes
  - *Cache* lookup results to avoid future lookups
- Client-side lookup cache
  - Lookups give: **`IP Address ⇒ Key Range`**
  - If key in a cached **`Key Range`**, avoid lookup

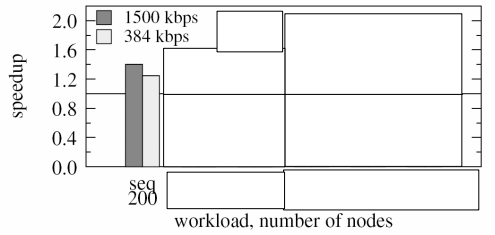
18

Performance Evaluation

- How much faster are tasks in D2 than Traditional?
- Deploy real implementation on Emulab
  - Same infrastructure DHT scenario as before
  - Emulate world-wide network topology
  - All DHTs compared use lookup cache
- Task replay modes:
  - Sequential: accesses dependent on each other
    - E.G., 'make'
  - Parallel: accesses not dependent on each other
    - E.G., 'cat *'

19

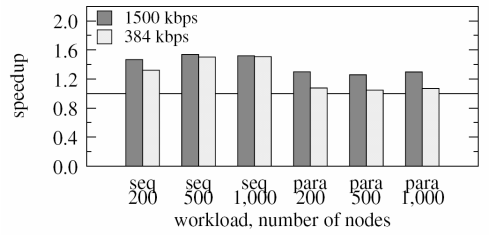
Performance Speedup



How much faster is D2 vs. Traditional?

20

Performance Speedup



How much faster is D2 vs. Traditional-**File**?

21

Questions

- Can task locality be maintained simply?
- Does locality outweigh parallelism?
- Can load balance be maintained cheaply?

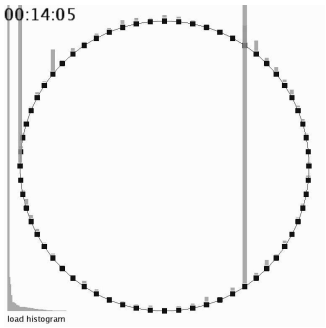
22

Technique #3: Dynamic Load Balancing

- Why?
  - Object placement is no longer uniform
  - Random node IDs ⇒ imbalance
- Simple dynamic load balancing
  - Karger [IPTPS'04], Mercury [SIGCOMM'04]
  - Fully distributed
  - Converges to load balance quickly

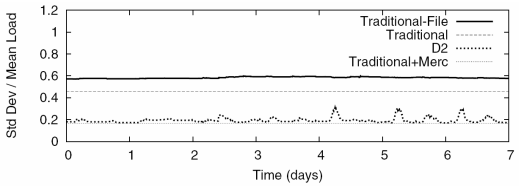
23

Dynamic Load Balancing



24

Load Balance



How balanced is storage load over time?

Load Balancing Overhead

Day	1	2	3	4	5	6	Total
Writes	61	71	142	114	109	123	620 MB/node
Load Balance	0	18	65	60	71	93	307

$$\frac{\text{Load Balance}}{\text{Writes}} \approx 50\% \approx \text{cost of 1 or 2 more replicas}$$

How much data is transferred to balance load?

Conclusions

- Can task locality be maintained simply?
  - **Yes**: Namespace locality enough to improve availability by an order of magnitude
- Does locality outweigh parallelism?
  - **Yes**: Real workloads observe 30-100% speedup
- Can load balance be maintained cheaply?
  - Maintain Balance? **Yes**: better than traditional
  - Cheaply? **Maybe**: 1 byte for every 2 bytes written

Defragmentation: a useful DHT technique

Thank You.

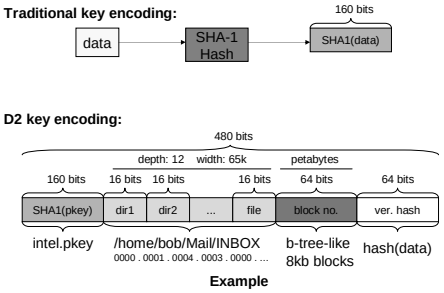
Backup Slides

- Or, more graphs than anyone cares for…

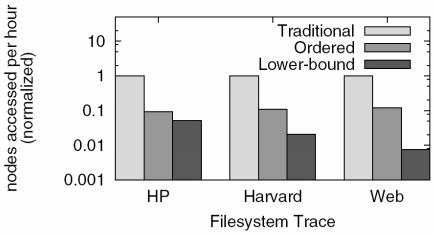
Related Work

- MOAT [Yu et al., NSDI '06]
  - Only studied availability
  - No mechanism for load balance
  - No evaluation of file systems
- SkipNet [Harvey et al., USITS '03]
  - Provides administrative locality
  - Can't do both locality + global load balance
- Mercury [Bharambe et al., SIGCOMM '04]
  - Dynamic load balancing for range queries
  - D2 uses Mercury as the routing substrate
  - D2 has independent storage layer, load balancing optimizations (see paper)

Encoding Object Names

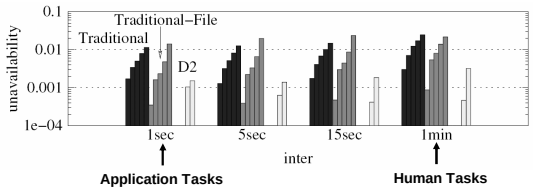


DHT Nodes Accessed



How many nodes does a user access?

Task Unavailability



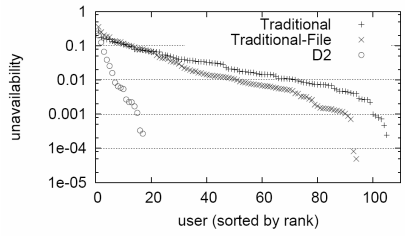
What fraction of tasks fail?

Task Unavailability

	mean objects		mean nodes		
inter	block	file	block	file	D2
1sec	63	10	10	6	2
5sec	91	15	11	8	2
15sec	128	22	14	10	3
1min	237	38	23	16	4

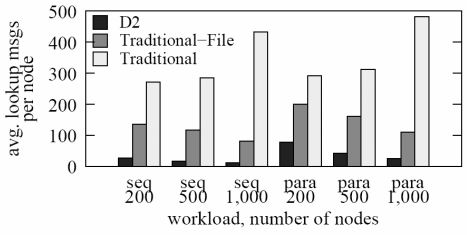
How many nodes are accessed per task?

Task Unavailability (Per-User)



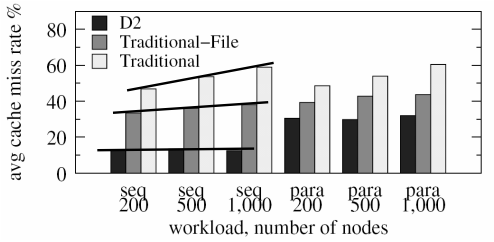
What fraction of each user's tasks fail?

DHT Lookup Messages



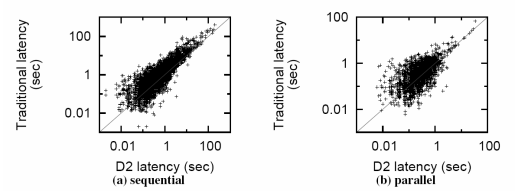
How many lookup messages are used?

Lookup Cache Utilization



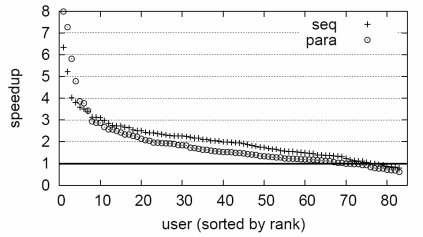
How many lookups miss the cache?

Performance Speedup (Per-Task)



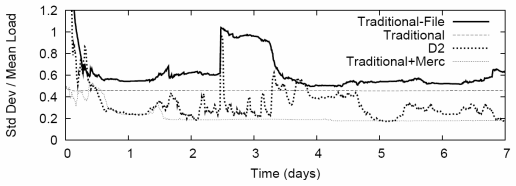
How much faster is D2 vs. Traditional?

Performance Speedup (Per-User)



How much faster is D2 vs. Traditional?

Load Balance (Webcache)



How balanced is storage load over time?

Load Balancing Overhead

Day	1	2	3	4	5	6	Total
Harvard W_i	61	71	142	114	109	123	620
Harvard L_i	0	18	65	60	71	93	307
Webcache W_i	353	32	36	398	428	355	1602
Webcache L_i	247	148	45	582	425	413	1860

How much data is transferred to balance load?