logistics

piazza is active now

paper discussions will start as of Feb 25

papers to be released next week

r. projects goal: lock topic after mid Feb
how to prepare slides

no bullets  2 colors  big text  connection from slide to slide
one message per slide  images  use animation for examples
how to prepare slides

no bullets    2 colors    big text    connection from slide to slide
one message per slide    images    use animation for examples

1) prepare slides, 2) meet with Stratos the week before
There is no such thing as a wrong question/answer!!!!

interaction: in and out of class
Research/Systems Intro

~3-4 lectures: Periodic Table, Calculator, NoSQL. Then Statistics, Neural Networks
How many and which structures are possible?
How many and which structures are possible?

Can we compute performance w/o coding?
NoSQL Key-value Stores

- RocksDB
- BigTable
- MongoDB
- SQLite
- Amazon DynamoDB
- Cassandra
- Apache HBase

Applications:
- machine learning
- social media
- smart homes
- web browsers
- phones
- web-based apps
- security
- health devices
- graphs
- analytics

Data Structures:
- b-tree
- lsm-tree
- log+index
insert (key-value)
insert (key-value)
insert (key-value)
DISK
MEMORY
Level 2
insert (key-value)

buffer

Level 1

Level 2

Level 3

...

Level N

MEMORY

DISK
insert (key-value)

buffer

Level 1

Level 2

Level 3

... 

Level N

MEMORY

DISK

leveled
tiered

sorted
insert (key-value)

buffer

Level 1

Level 2

Level 3

...

Level N

MEMORY

DISK

pages

SSTables

leveled

tiered

sorted
[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted

[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted

[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted

[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted

[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted

[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted

[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted

[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted

[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted

[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted

[1,0,0,1,1,1] hash fun.

bloom filters

[min-max] /page

fence pointers

buffer

Level 1

Level 2

Level 3

Level N

MEMORY

DISK

pages

SSTables

leveled
tiered

sorted
get (key)

buffer

Level 1

Level 2

Level 3

... 

Level N

bloom filters

fence pointers

[1,0,0,1,1,1] hash fun.

[min-max] /page

Level 2

Level N

SSTables
pages
tiered

leveled

sorted

MEMORY

DISK

get (key)
get (key)

buffer

Level 1

Level 2

Level 3

Level N

bloom filters

fence pointers

[1,0,0,1,1,1]
hash fun.

[min-max]
pages

Level 1

Level 2

Level 3

Level N

[1,0,0,1,1,1]
hash fun.

[min-max]
pages

bloom filters

fence pointers

Level 1

Level 2

Level 3

Level N

DASlab
© Harvard SEAS
get (key)

Level 1

Level 2

Level 3

... Level N

bloom filters
fence pointers
[min-max] /page

[1,0,0,1,1,1] hash fun.

[1,0,0,1,1,1]

memory

disk

pages

SSTables

leveled

sorted

tiered

hash fun.

fence pointers

[1,0,0,1,1,1]
bloom filters
fence pointers
[1,0,0,1,1,1]
hash fun.
[min-max]/page
[1,0,0,1,1,1]
hash fun.
[min-max]/page

Level 1

Level 2

Level 3

... ...

Level N

MEMORY
DISK
pages
SSTables
tiered
leveled
sorted

buffer
bloom filters
fence pointers

[1,0,0,1,1,1] hash fun.
[min-max] /page

hash fun.
fence pointers

[1,0,0,1,1,1]

size ratio
merge policy
filters bits per entry
size of buffer/cache
internal k-v layout

MEMORY
DISK

glevels
page
SSTables
tiered

teveled

sorted

Level 1

Level 2

Level 3

Level N

size ratio
merge policy
filter bits per entry
size of buffer/cache
internal k-v layout
size ratio
merge policy
filters bits per entry
size of buffer/cache
internal k-v layout
DOMAIN?

- size ratio
- merge policy
- filters bits per entry
- size of buffer/cache
- internal k-v layout
BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: Optimal Navigable Key-Value Store

@SIGMOD2017
BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: Optimal Navigable Key-Value Store

@SIGMOD2017
BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: Optimal Navigable Key-Value Store

@SIGMOD2017

bits per entry: fixed per run

buffer

Level 1

Level 2

...  

Level N
 worst lookup cost: 
sum of false positive rates

bits per entry:
fixed per run

buffer

Level 1

Level 2

... 

Level N
BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: Optimal Navigable Key-Value Store

@SIGMOD2017

the same memory budget is more impactful at smaller levels

bits per entry: fixed per run

buffer

Level 1

Level 2

... 

Level N
BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: Optimal Navigable Key-Value Store

the same memory budget is more impactful at smaller levels

buffer

Level 1

Level 2

Level N

...
BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: Optimal Navigable Key-Value Store

the same memory budget is more impactful at smaller levels

buffer

Level 1

Level 2

...  

Level N

lookup cost

update cost

WiredTiger

Cassandra, HBase

RocksDB, LevelDB

monkey

BITS PER ENTRY IN FILTERS: OPTIMIZED OUT

Monkey: Optimal Navigable Key-Value Store

the same memory budget is more impactful at smaller levels

buffer

Level 1

Level 2

...  

Level N

lookup cost

update cost

WiredTiger

Cassandra, HBase

RocksDB, LevelDB

monkey
the same memory budget is more impactful at smaller levels

Level 1

Level 2

Level N

---

Look up latency (ms)

LevelDB

Monkey

≈ 0.2 I/Os per lookup

uniform, zero result, point queries, entry size=1KB
MERGE POLICY: SHOULD BE TUNED

Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store
MERGE POLICY: SHOULD BE TUNED

Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store

buffer

Level 1

Level 2

...

Level N
MERGE POLICY: SHOULD BE TUNED

Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store

merge policy: fixed across levels

helps reads

LeveN 2

helps writes

Level N
MERGE POLICY: SHOULD BE TUNED

Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store

merge policy: fixed across levels

Level 1

Level 2

...  

Level N
MERGE POLICY: SHOULD BE TUNED

Doostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store

merging small levels does not help that much (point, range, space)

merge policy: fixed across levels

buffer

Level 1

Level 2

... 

Level N
merging small levels does not help that much (point, range, space)
merging small levels does not help that much (point, range, space)
MERGE POLICY: SHOULD BE TUNED

Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store

merging small levels does not help that much (point, range, space)
 monthly savings $$ vs Google LevelDB

Amazon Cloud (North America)
25% monthly savings $\$\$ vs Google LevelDB

- $10K at 20TB
- $500K at 20PB

Amazon Cloud (North America)
25% monthly savings $" vs Google LevelDB

Data Size

$10K

20TB 200TB 2PB 20PB

Amazon Cloud (North America)
25% monthly savings $\$\$\$\$ vs Google LevelDB

Data Size

$10K

20TB 200TB 2PB 20PB

$500K

Amazon Cloud (North America)
a type of project:

Pick problem area (NoSQL, Structure Class, Neural Network)

Figure out design principles

Study optimal values for one of them

Demonstrate benefits with respect to problem area
LSM-trees

- Size ratio
- Merge policy
- Filters bits per entry
- Size of buffer/cache
- Internal k-v layout
unified design space
POSSIBLE NODE DESIGNS
POSSIBLE NODE DESIGNS  POSSIBLE STRUCTURES

- sorted zone map
- bloom filter bits
- link
- children
- layout

@SIGMOD18
POSSIBLE NODE DESIGNS

POSSIBLE STRUCTURES

- Trie
- Array
- Skip-List
- Linked-List
- Sorted Array
- Hash-Table
BIG DATA SYSTEMS
NoSQL | Neural Networks | SQL | Graph | Data Science