USC University of Southern California

The Implementation of B+ Trees Within MongoDB

CSCI 550

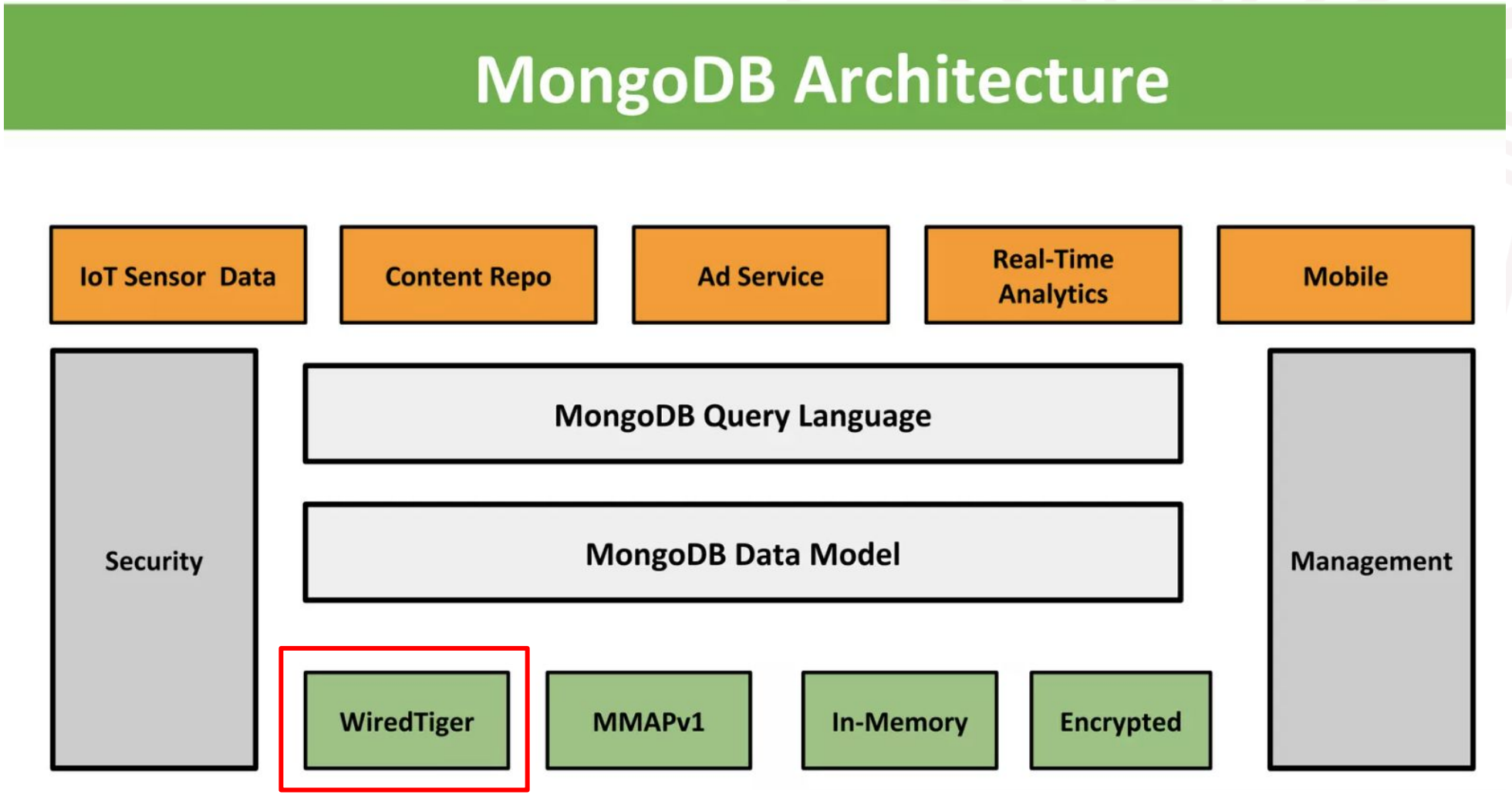
Team 2 -- Changxun Li, Haoxiang Geng, Yueqin Li

USC

1. Introduction

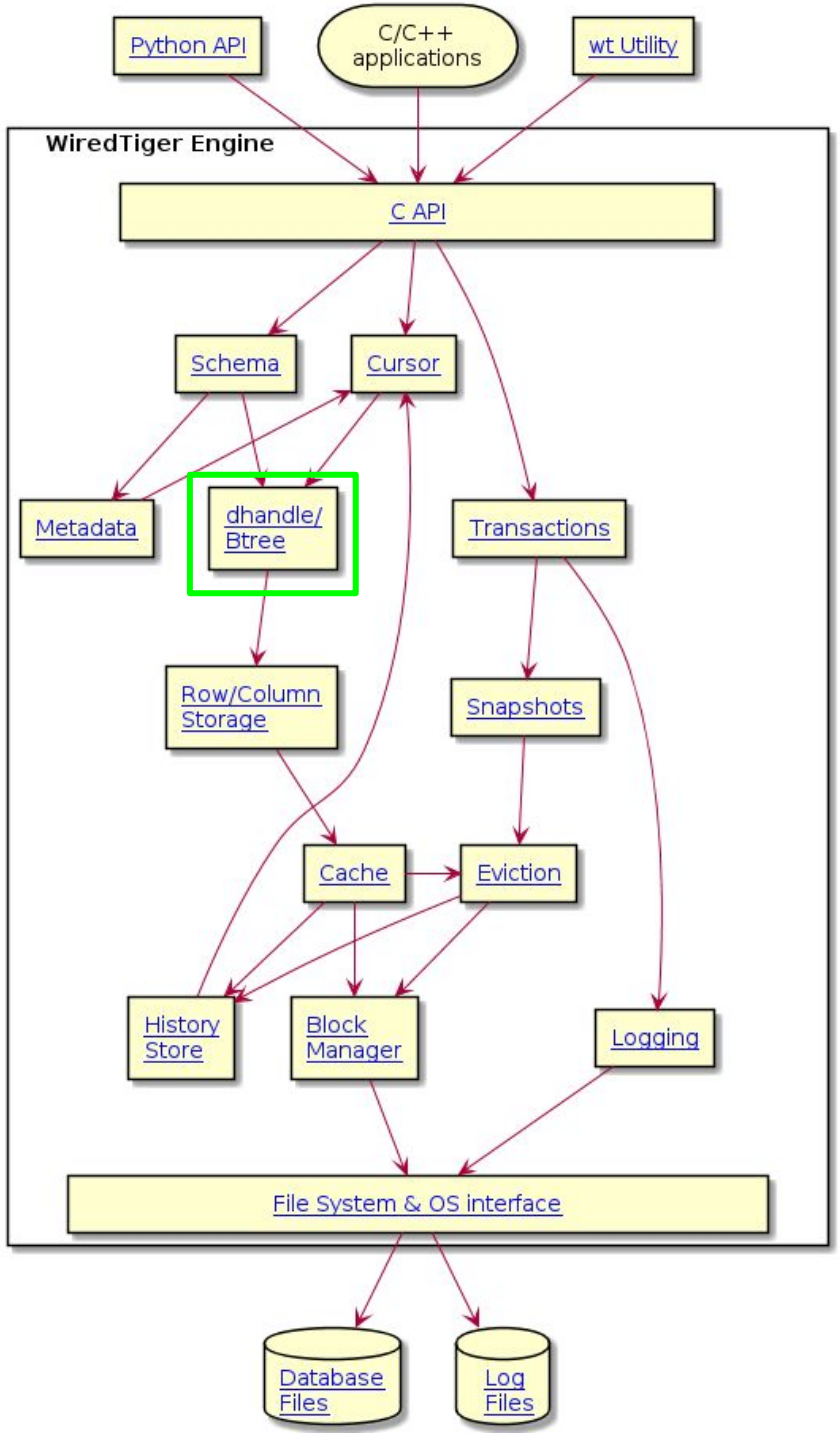
USC

MongoDB


MongoDB Architecture
IoT Sensor Data
Content Repo
Ad Service
Real-Time Analytics
Mobile
Security
MongoDB Query Language
MongoDB Data Model
Management
WiredTiger
MMAPv1
In-Memory
Encrypted

WiredTiger (WT) :

MongoDB's Storage Engine



USC

Potential Problems & Limitations

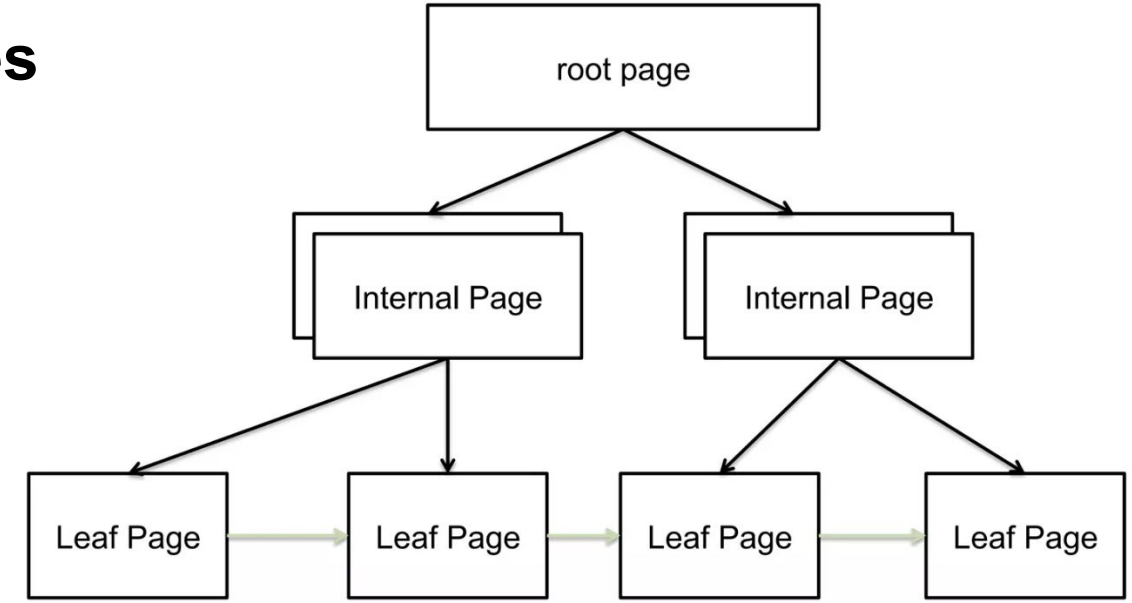
- Fun fact: WT only support B-Tree indexing! – suitable for OLTP, but what about OLAP?
- But B-Tree indexing may not be as efficient due to:
  - Upwards traversal through parents to find siblings
  - Worse performance for sequential and range scans

Research Questions

- Why not use B+ Tree which is supposed to be better?
- What will happen if it uses B+ Tree instead?

Proposed solution

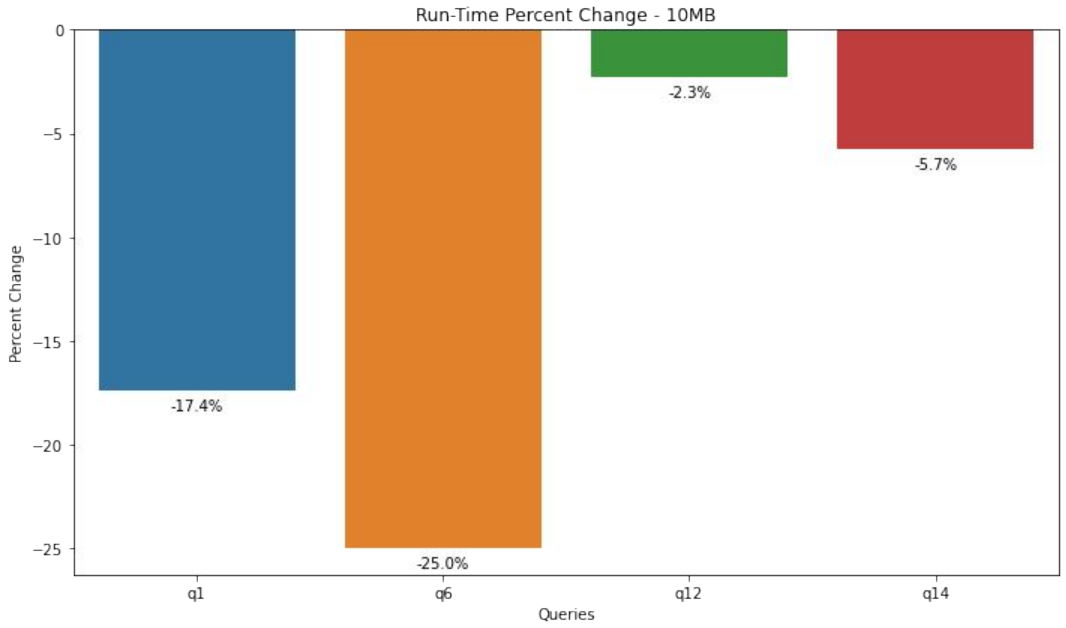
- Full B+ Tree – **link list all leaf nodes**
- **Traverse** between leaves **directly**
- **Faster** sequential and range **scans**

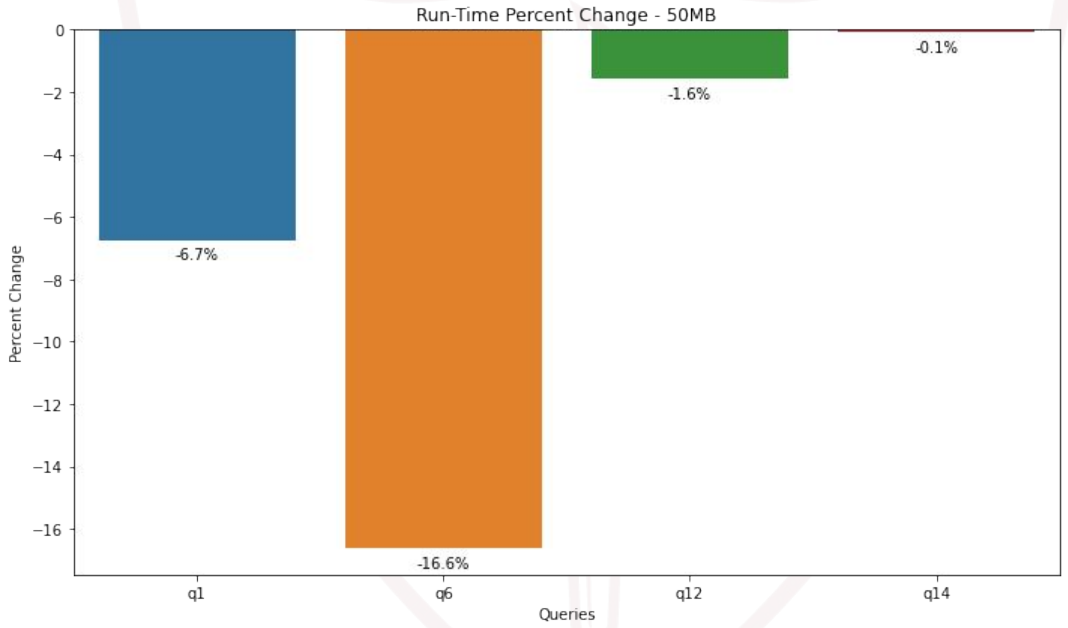


USC

Experimental Evaluation

- 1%~25% run-time reduction for all selected TPC-H queries





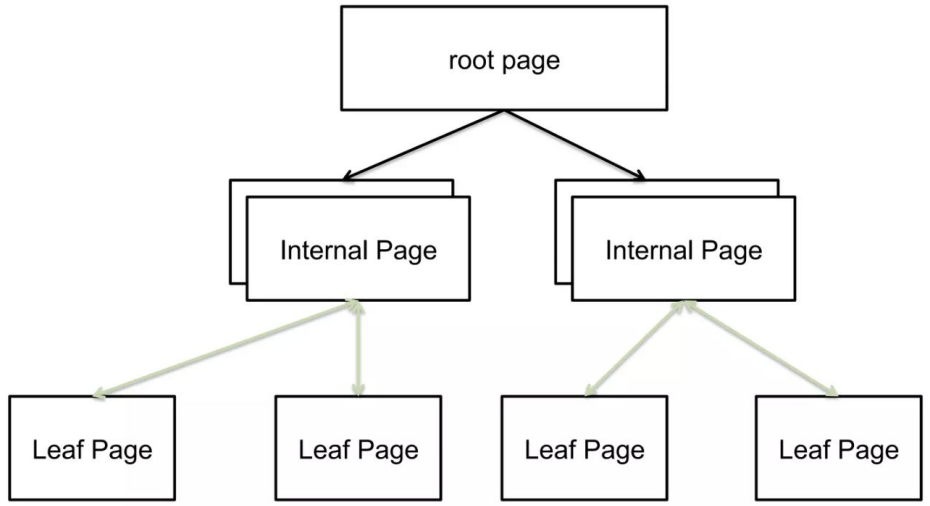
USC

2. Our Approach - B+ Tree

USC

WiredTiger Exploration

- Internal & leaf pages
- Internal pages are guides/directories
  - Parent nodes direct searches towards the correct child nodes
- Leaf pages hold data

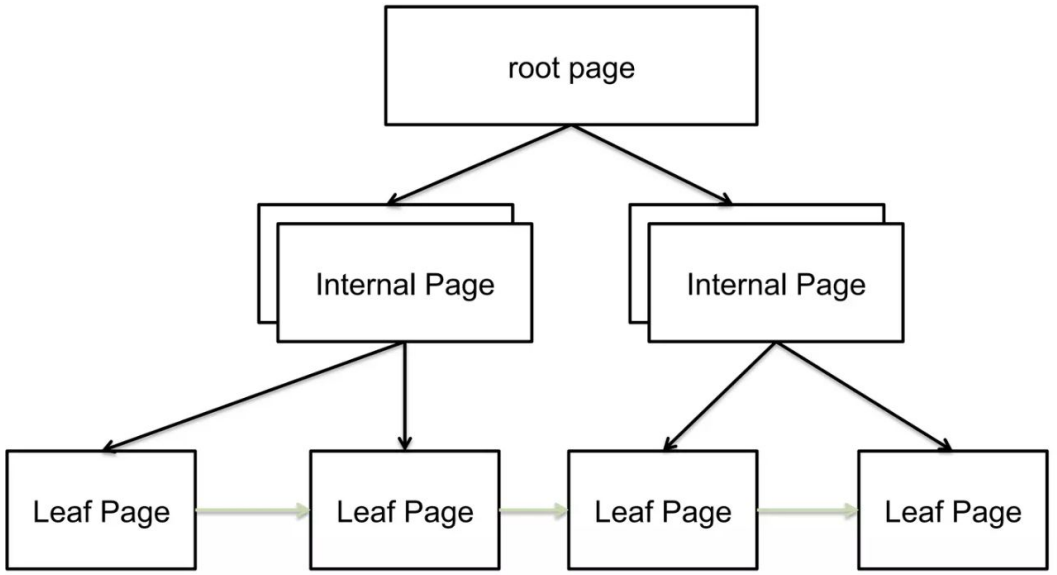


WiredTiger B-tree

USC

Traditional B+ Tree

- All leaf nodes are linked together sequentially, allowing for efficient ordered data retrieval and range scans

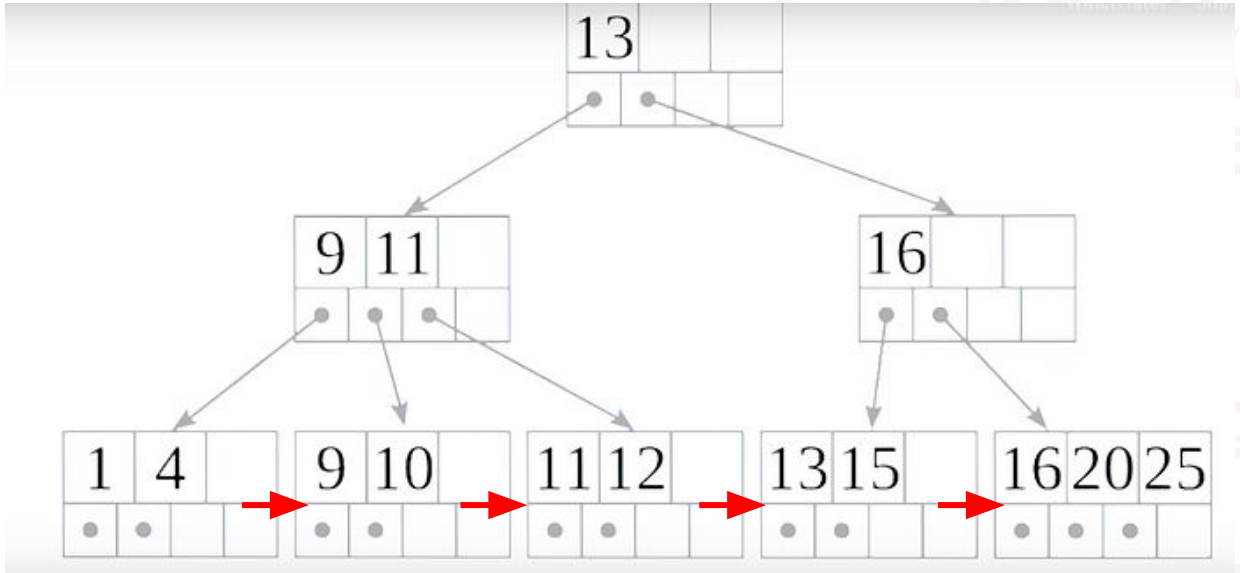


Traditional B+ tree

USC

Our Objective

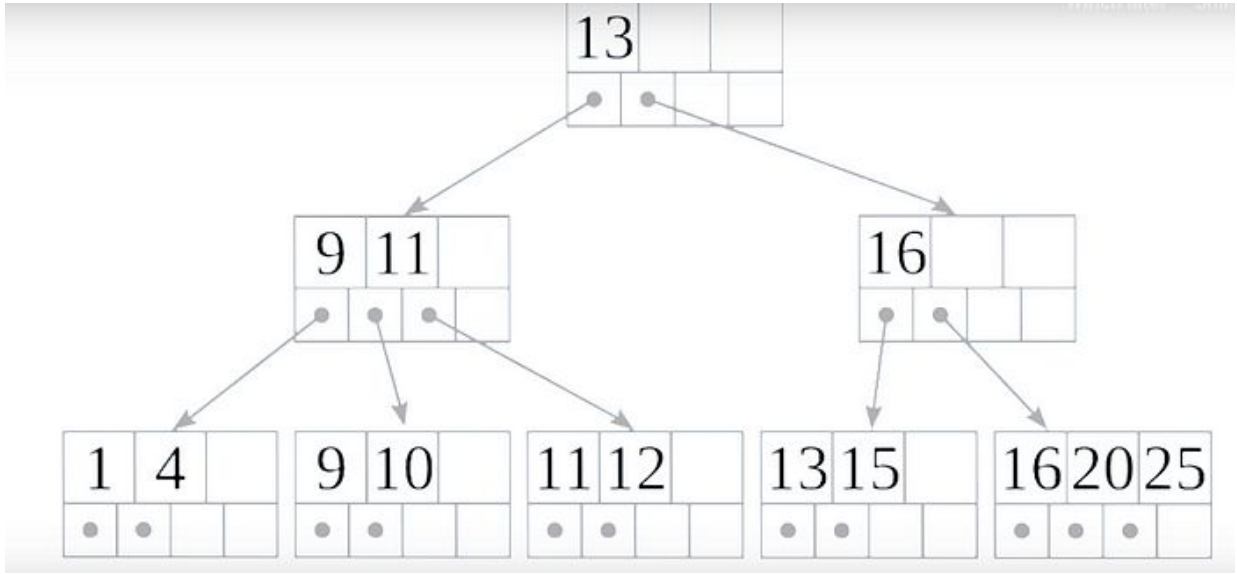
- Modify the B-tree structure to better support OLAP transactions
- Use the linked list of B+ tree leaf nodes to enhance the efficiency of sequential scans



USC

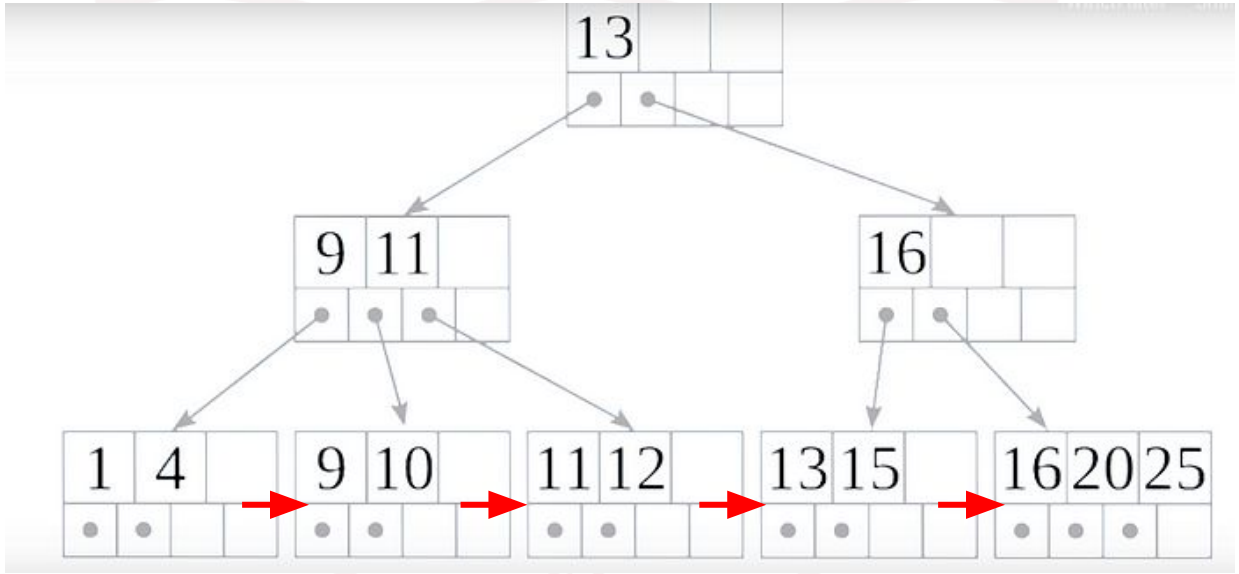
Our Solution

Original WT_Btree



Utilizing ascending and descending operations

Our Solution



Utilizing next pointer between leaf nodes

USC

3. Experiments

USC

Setup

- 2021 M1 ARM Macbook Pro
- macOS 14.4.1
- 16 GB of RAM
- MongoDB 6.0 + Wiredtiger v10.0.2
- TPC-H benchmark + Q1, Q6, Q12, Q14

USC

TPC-H Q1

```
## This query reports the amount of business that was billed, shipped, and returned

pipeline = [
  {
    "$match": {
      "_id": {"$gte":ObjectId(
    }
  },
  {
    "$group": {
      "_id": {"l_returnflag":
      "sum_qty": {"$sum": "$L_
      "sum_base_price": {"$sum
      "sum_disc_price": {"$sum
      "sum_charge": {"$sum": {
      "avg_qty": {"$avg": "$L_
      "avg_price": {"$avg": "$
      "avg_disc": {"$avg": "$L
      "count_order": {"$sum":
    }
  },
  {
    "$sort": {"_id.l_returnfla
  }
]

start_time = datetime.now()

results_1 = deals_db.aggregate

end_time = datetime.now()
execution_time = end_time - st

for result_1 in results_1:
  print(result_1)

print(f"Query executed in: {ex
```

TPC-H Q6

```
## Quantifies the amount of revenue increa
## certain company-wide discounts in a giv

pipeline = [
  {
    "$match": {
        "L_SHIPDATE": {
        "$gte": datetime(1994, 1, 1),
        "$lt": datetime(1995, 1, 1)
      },
      "L_DISCOUNT": {"$gte": 0.05, "$lte":
      "L_QUANTITY": {"$lt": 24}
    }
  },
  {
    "$group": {
      "_id": None,
      "revenue": {"$sum": {"$multiply": ["
    }
  }
]
start_time = datetime.now()
result_6 = deals_db.aggregate(pipeline).ne
print('Revenue Increase:', result_6['reven

end_time = datetime.now()
execution_time = end_time - start_time

print(f"Query executed in: {execution_time
```

TPC-H Q12

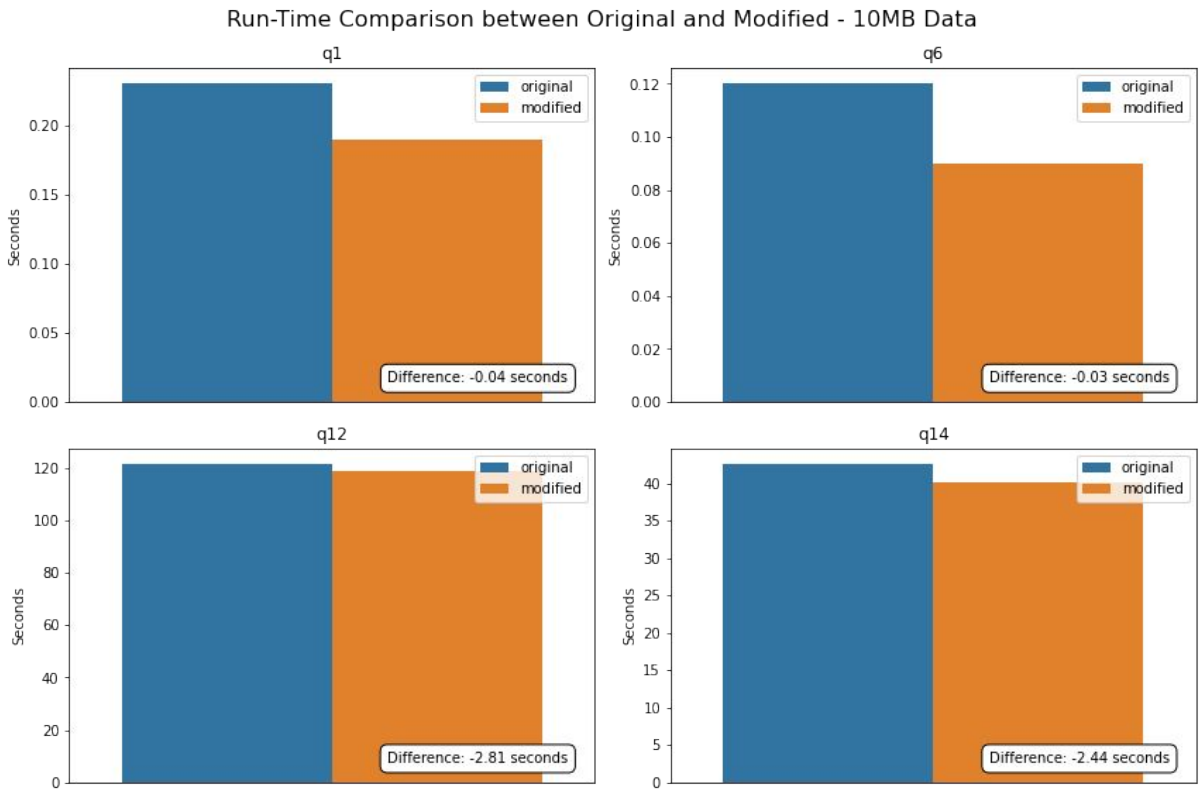
```
# Determines the number of people chooisng cheaper or more expensive shipping methods

pipeline = [
  {
    "$match": {
      "L_SHIPMODE": {"$in": ["MAIL",
      "L_RECEIPTDATE": {"$gte": date
      "_id": {"$gte":ObjectId("00000
      "L_SHIPMODE": {"$in": ["MAIL",
    }
  },
  {
    "$lookup": {
      "from": "order",
      "localField": "L_ORDERKEY",
      "foreignField": "O_ORDERKEY",
      "as": "order_info"
    }
  },
  {
    "$unwind": "$order_info"
  },
  {
    "$group": {
      "_id": "$L_SHIPMODE",
      "high_line_count": {
        "$sum": {
          "$cond": {
            "if": {"$in": ["$order_i
            "then": 1,
            "else": 0
          }
        }
      },
      # Alternative for low_line_cou
      "low_line_count": {
        "$sum": {
          "$cond": {
            "if": {"$not": {"$in": [
            "then": 1,
            "else": 0
          }
        }
      }
    }
  },
  {
    "$sort": {"_id": 1}
  }
]

start_time = datetime.now()
results_12 = deals_db.aggregate(pipe
end_time = datetime.now()
execution_time = end_time - start_ti

print(f"Query executed in: {executio
for result_12 in results_12:
  print(result_12)
```

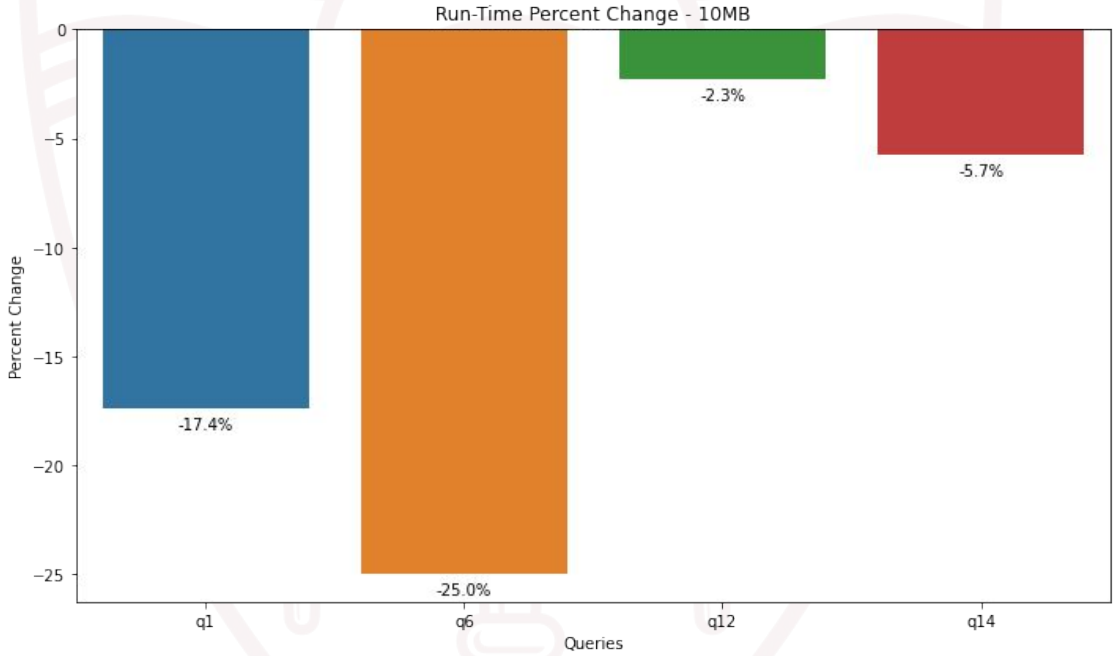
TPC-H Q14

```
## Monitors the market response to a promotion such as TV advertisements or a special campaign

pipeline = [
  {
    "$match": {
      "_id":{
        "$gte": ObjectId("000000000000000000111111")
      },
      "L_SHIPDATE": {
        "$gte": datetime(1995, 9, 1),
        "$lt": datetime(1995, 10, 1)
      }
    }
  },
  {
    "$lookup": {
        "from": "part",
        "localField": "L_PARTKEY",
        "foreignField": "P_PARTKEY",
        "as": "part_info"
    }
  },
  {
    "$unwind": "$part_info"
  },
  {
    "$group": {
      "_id": None,
      "promo_revenue": {
        "$sum": {
          "$cond": {
            "if": {"$regexMatch": {"input": "$part_info.P_TYPE", "regex": "^PROMO"}},
            "then": {"$multiply": ["$L_EXTENDEDPRICE", {"$subtract": [1, "$L_DISCOUNT"]}]},
            "else": 0
          }
        }
      },
      "total_revenue": {"$sum": {"$multiply": ["$L_EXTENDEDPRICE", {"$subtract": [1, "$L_DISCOUNT"]}]}}
    }
  },
  {
    "$project": {
      "_id": 0,
      "promo_revenue": {"$multiply": [100, {"$divide": ["$promo_revenue", "$total_revenue"]}]}
    }
  }
]
start_time = datetime.now()

result_14 = deals_db.aggregate(pipeline).next()

end_time = datetime.now()
execution_time = end_time - start_time

print(f"Query executed in: {execution_time.total_seconds()} seconds")
print('Promo Revenue Percentage:', result_14['promo_revenue'])
```

USC

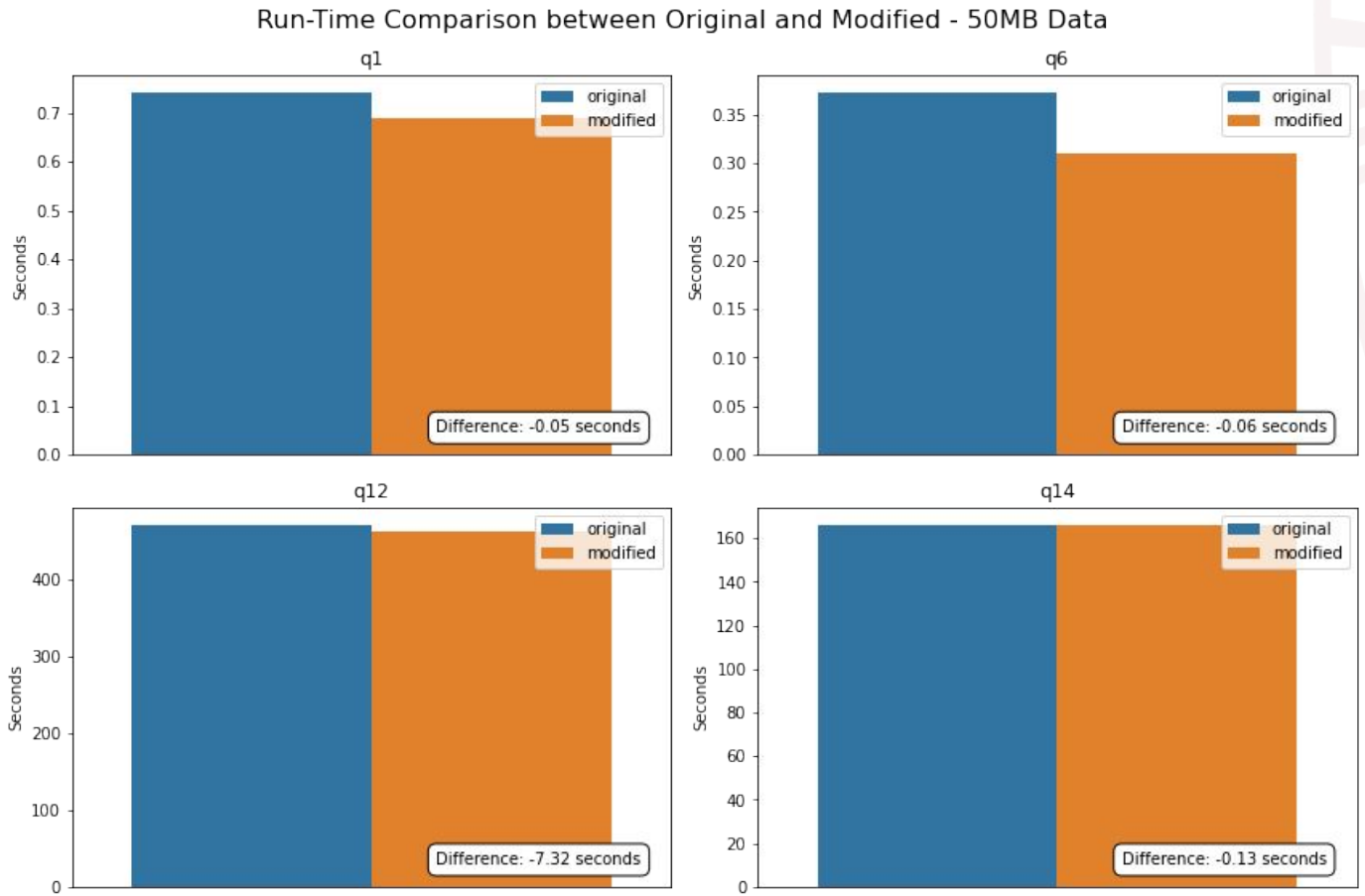
Evaluation & Comparison - 10MB

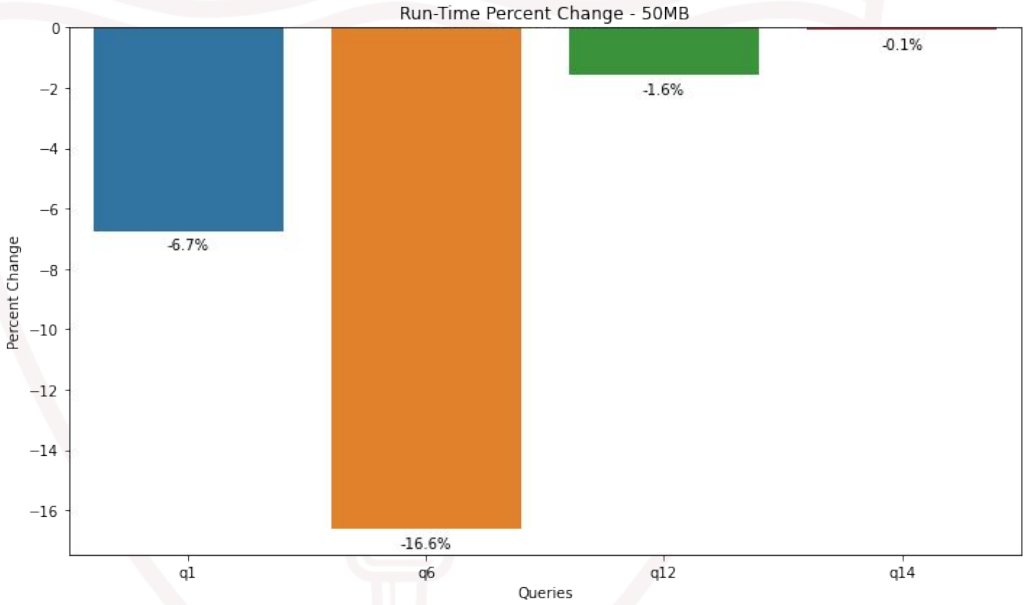




USC

Evaluation & Comparison - 50MB





USC

Analysis

- WiredTiger's B-Tree typically maintain a shallow depth, even for large datasets, due to the high fan-out of each node

- Using ascending and descending operations won't introduce much overhead

B+tree structure for 10MB data (0.1 million rows)

```
Page Address: 0x6000030f0bd0, Type: Internal (Row-Store), Level: 0
  Child Page Addresses:
    0x6000030fcbd0
    0x6000030fca80
    0x6000030ec690
    0x6000030f0ee0
    0x6000030f0f50
    0x6000030fcaf0
  Page Address: 0x6000030fcbd0, Type: Leaf (Row-Store), Level: 1
  Page Address: 0x6000030fca80, Type: Leaf (Row-Store), Level: 1
  Page Address: 0x6000030ec690, Type: Leaf (Row-Store), Level: 1
  Page Address: 0x6000030f0ee0, Type: Leaf (Row-Store), Level: 1
  Page Address: 0x6000030f0f50, Type: Leaf (Row-Store), Level: 1
  Page Address: 0x6000030fcaf0, Type: Leaf (Row-Store), Level: 1
```

Further Thoughts

- B+ tree should significantly outperforms B-tree in range scan
  - Utilizing pointers between leaf node

- Might not be suitable for all cases
  - WiredTiger maintains a shallow-depth B-tree

- Somehow explains why MongoDB utilizes B-tree for indexing

Thank You!

USC

USC University of Southern California

Q&A

USC