DDBMS

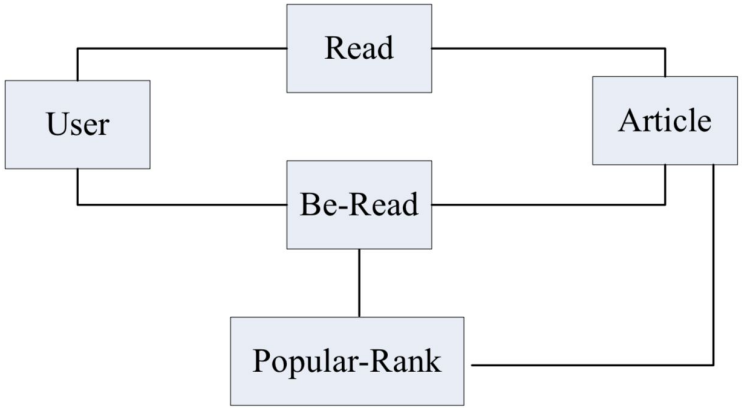
Final presentation

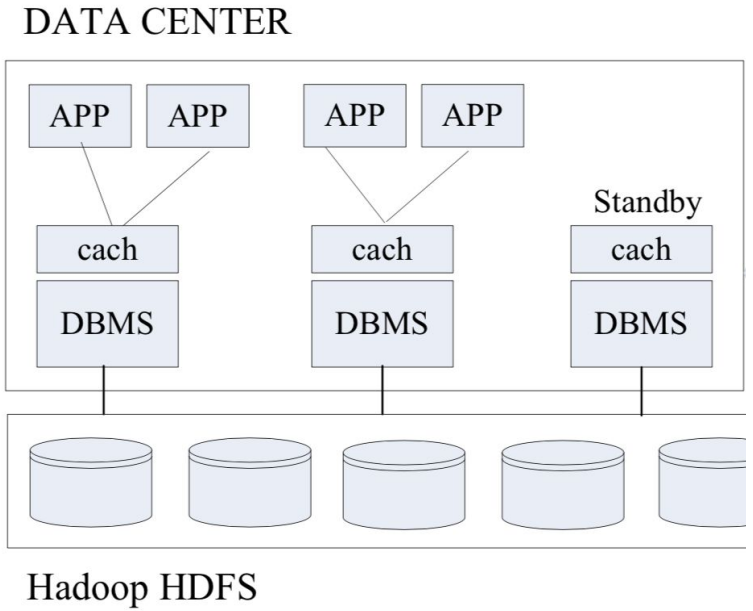
YanZhao Chen, Péter Garamvölgyi

outline

- problem overview
- design overview
- database design (mongoDB, pymongo)
- blob storage (HDFS/Hadoop, hdfs.py)
- web API (flask) + caching (redis)
- monitoring (mongoDB cloud monitoring)
- challenges & future work

problem overview





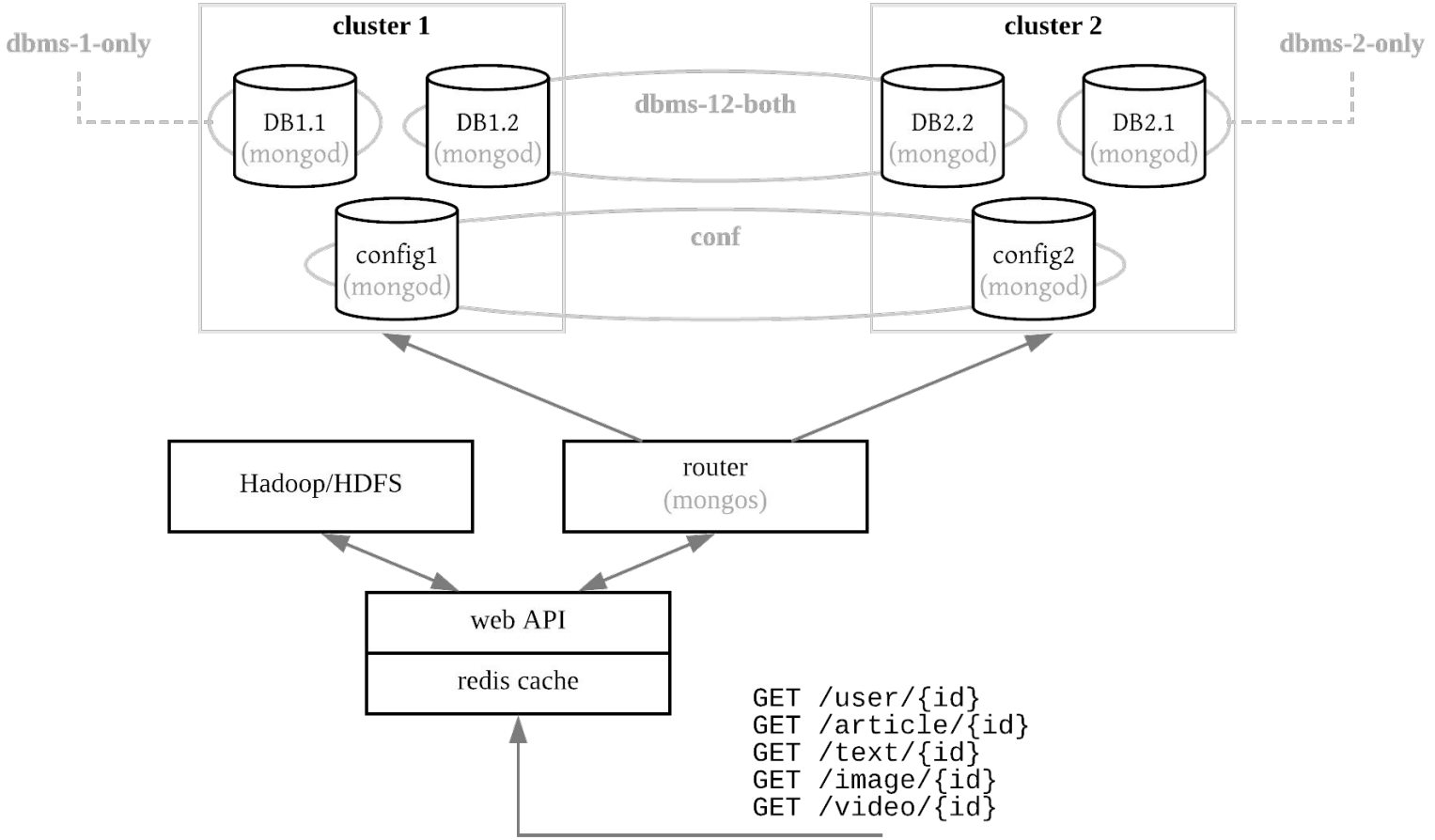
design overview

- philosophy:

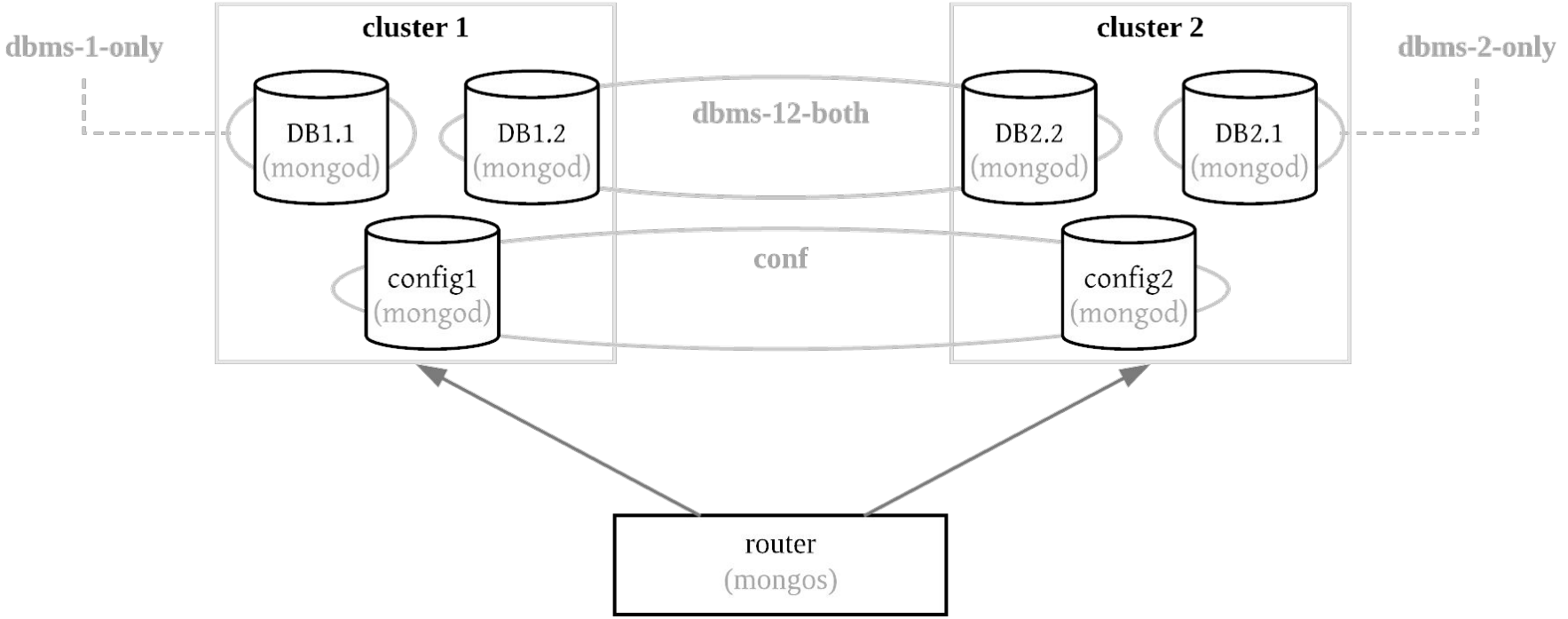
  “don’t reinvent the wheel”

- mongoDB has built-in support for
  - partitioning (sharding)
  - replication
  - complex, distributed query processing
  - monitoring

design overview



database design

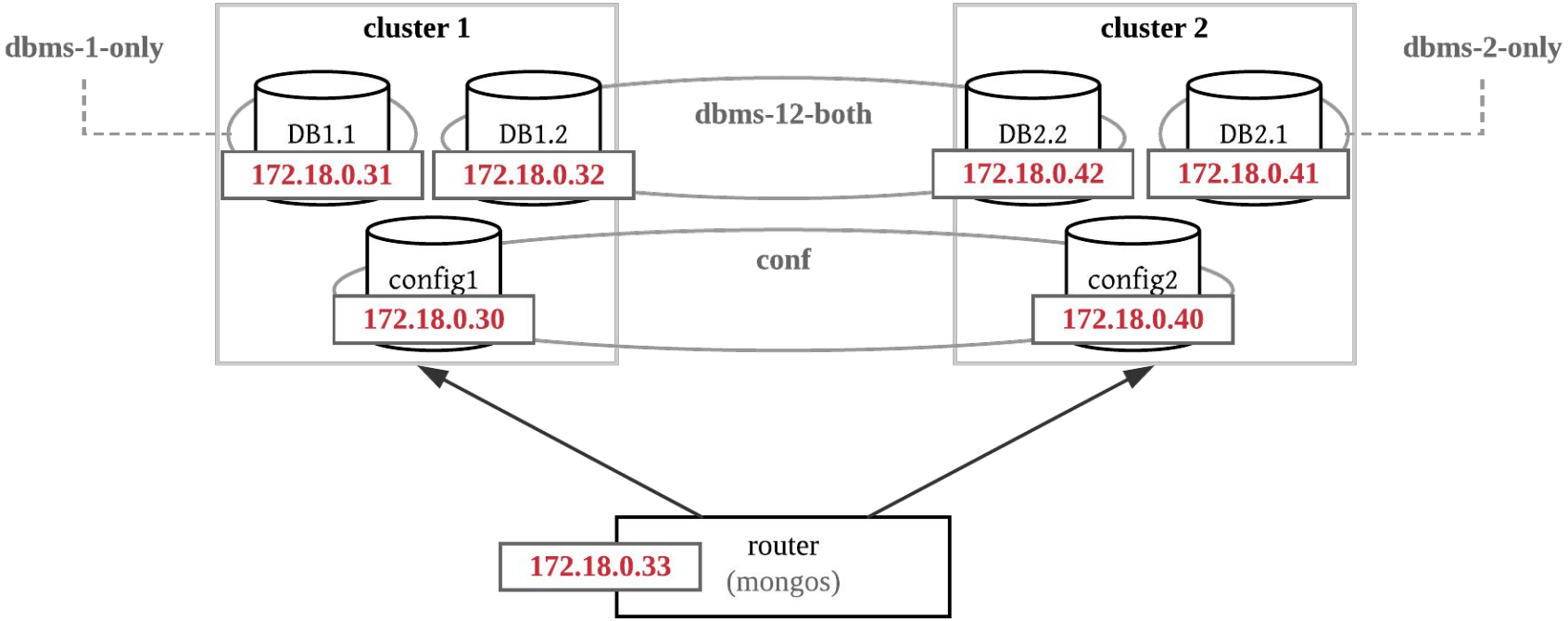


database design - docker-compose

```
networks:
  mongonet:
    driver: bridge
    ipam:
      driver: default
      config:
      - subnet: 172.18.0.0/16

  dbms11:
    container_name: dbms11
    image: mongo
    networks:
      mongonet:
        ipv4_address: 172.18.0.31
    volumes:
      - ./db/dbms11:/data/db
    command: "mongod --shardsvr --replSet dbms1-only --bind_ip 172.18.0.31 --port 27017"
```

database design



database design - replication

```
$ docker run -it --net cluster_mongonet mongo mongo mongodb://172.18.0.30:27017
> rs.initiate(
  {
    _id: "config",
    configsvr: true,
    members: [
      { _id : 0, host : "172.18.0.30:27017" },
      { _id : 1, host : "172.18.0.40:27017" }
    ]
  }
)
```

database design - replication

- expansion at the DBMS-level allowing a new DBMS server to join ✓
- dropping a DBMS server at will ✓

database design - sharding

```
$ docker run -it --net cluster_mongonet mongo mongo mongodb://172.18.0.33:27017
> sh.addShard("dbms1-only/172.18.0.31:27017")
> sh.addShard("dbms2-only/172.18.0.41:27017")
> sh.addShard("dbms12/172.18.0.32:27017")
> sh.status()
  ...
  shards:
        {  "_id" : "dbms1-only",  "host" : "dbms1-only/172.18.0.31:27017",
        "state" : 1 }
        {  "_id" : "dbms12",  "host" :
        "dbms12/172.18.0.32:27017,172.18.0.42:27017",  "state" : 1 }
        {  "_id" : "dbms2-only",  "host" : "dbms2-only/172.18.0.41:27017",
        "state" : 1 }
  ...

> sh.enableSharding("db")
```

database design - sharding

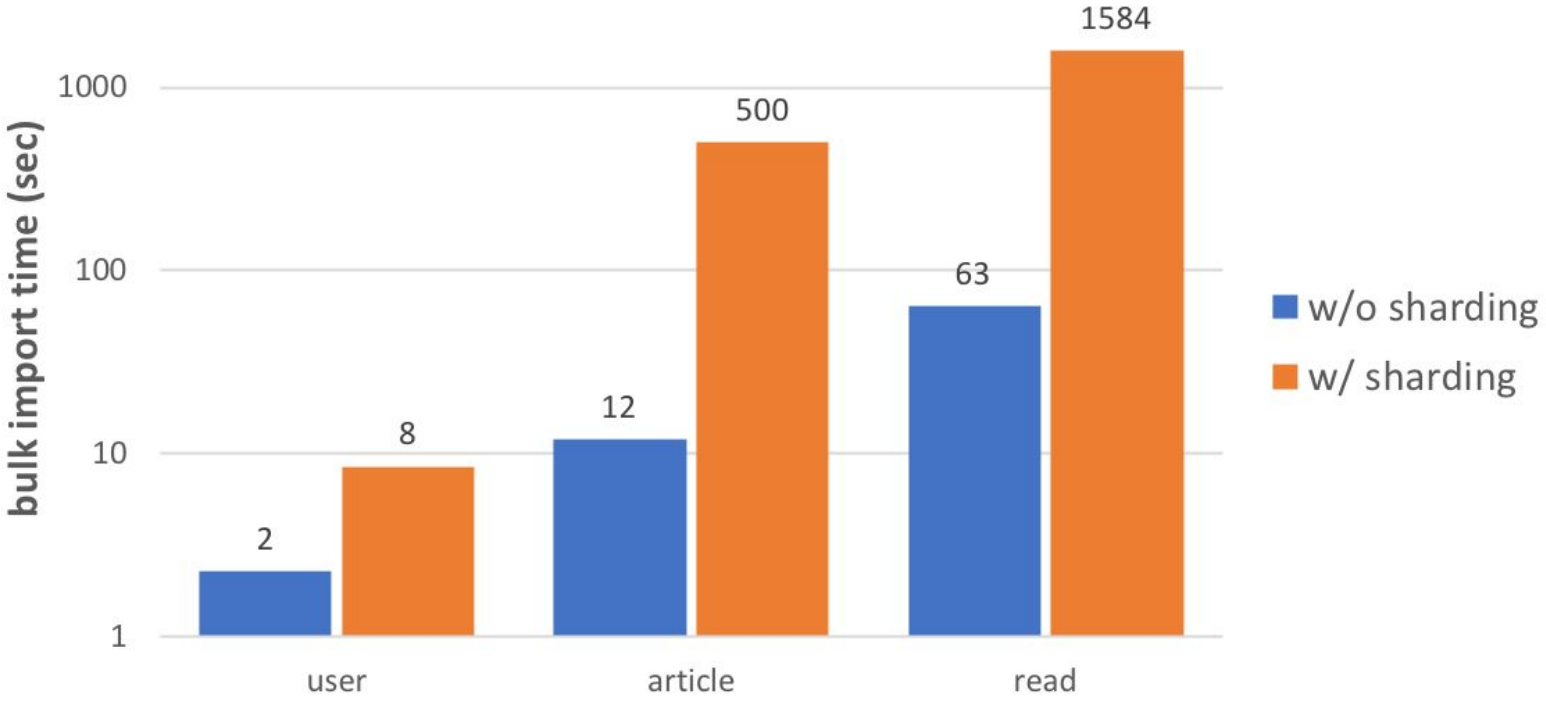
```
> sh.shardCollection("db.user", { region : 1 })
> sh.disableBalancing("db.user")

> sh.splitAt("db.user", { region: "Beijing" })
> sh.splitAt("db.user", { region: "Hong Kong" })
> sh.moveChunk("db.user", { region: "Beijing" }, "dbms1-only")
> sh.moveChunk("db.user", { region: "Hong Kong" }, "dbms2-only")
```

```
> sh.shardCollection("db.article", { category : 1 })
> sh.disableBalancing("db.article")

> sh.splitAt("db.article", { category: "science" })
> sh.splitAt("db.article", { category: "technology" })
> sh.moveChunk("db.article", { category: "science" }, "dbms12")
> sh.moveChunk("db.article", { category: "technology" }, "dbms2-only")
```

database design - sharding



database design - sharding

database design - collection generation (pymongo)

```
pipeline = [
    {
        "$group": {
            "_id": "$aid",
            "aid": { "$first": "$aid" },
            "category": { "$first": "$category" },
            "readNum"   : { "$sum": { "$toInt": "$readOrNot" }},
            "commentNum": { "$sum": { "$toInt": "$commentOrNot" }},
            "agreeNum"  : { "$sum": { "$toInt": "$agreeOrNot" }},
            "shareNum"  : { "$sum": { "$toInt": "$shareOrNot" }},
            "readUidList"   : { "$addToSet": { "$cond": [{ "$eq": ["$readOrNot"   , "1"] }, "$uid", "$noval"] } },
            "commentUidList": { "$addToSet": { "$cond": [{ "$eq": ["$commentOrNot", "1"] }, "$uid", "$noval"] } },
            "agreeUidList"  : { "$addToSet": { "$cond": [{ "$eq": ["$agreeOrNot"  , "1"] }, "$uid", "$noval"] } },
            "shareUidList"  : { "$addToSet": { "$cond": [{ "$eq": ["$shareOrNot"  , "1"] }, "$uid", "$noval"] } }
        }
    },
    { "$out": "be-read-temp" }
]
```

database design - collection generation (pymongo)

```
db["read"].aggregate([
    {
        "$project": {
            "aid": 1,
            "date": { "$dateToString": {
                "format": "%Y-%m-%d (day)",
                "date": { "$toDate": { "$toLong": "$timestamp" }}
            }},
            "readOrNot": { "$toInt": "$readOrNot" }
        }
    },
    {
        "$group": {
            "_id": { "date": "$date", "aid": "$aid" },
            "count": { "$sum": "$readOrNot" }
        }
    },
    { "$sort": { "_id.date": 1, "count": -1, "_id.aid": 1 }},
    {
        "$group": {
            "_id": "$_id.date",
            "top": { "$push": { "aid": "$_id.aid", "count" : "$count" }}
        }
    },
    { "$project": { "articleAidList": { "$slice": [ "$top", 5 ] }}},
    { "$addFields": { "temporalGranularity": "daily" }},
    { "$out": "popular-rank-daily" }
], allowDiskUse=True)
```

blob storage

- simply use HDFS/Hadoop single-node docker image and hdfs.py

```
hdfs_client.makedirs('/data/texts')
hdfs_client.makedirs('/data/images')
hdfs_client.makedirs('/data/videos')

hdfs_client.upload('/data/texts/84393add8c', './data/texts/84393add8c.txt')
hdfs_client.upload('/data/images/77af778b51', './data/images/77af778b51.jpg')
hdfs_client.upload('/data/videos/92a15e5a53', './data/videos/92a15e5a53.mp4')

db.article.update_many({}, { "$set" : {
    "text" : "84393add8c",
    "image": "77af778b51",
    "video": "92a15e5a53"
}})
```

web API (flask)

```
app = flask.Flask(__name__)

def retrieve_user(uid):
    user = db["user"].find_one({ "uid" : str(uid) })
    if user is not None:
        del user["_id"]
    return user

@app.route('/user/<int:uid>', methods=['GET'])
def user(uid):
    user = retrieve_user(uid)
    return flask.jsonify(user) if user is not None else ("User not found", 404)

app.run(debug=True)
```

caching (redis)

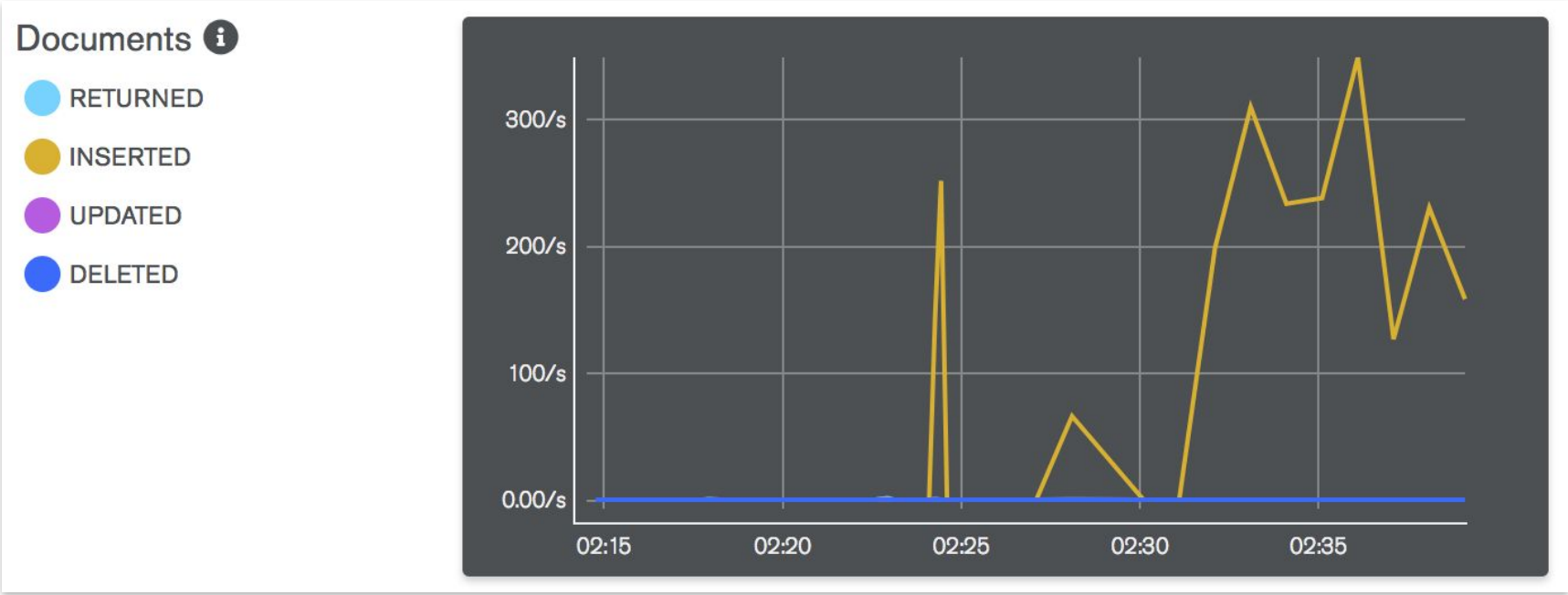
- simply use redis official docker image and flask_caching

```
cache = flask_caching.Cache(app, config={'CACHE_TYPE': 'redis'})

@cache.memoize(timeout=60)
def retrieve_user(uid):
    user = db["user"].find_one({ "uid" : str(uid) })
    if user is not None:
        del user["_id"]
    return user
```

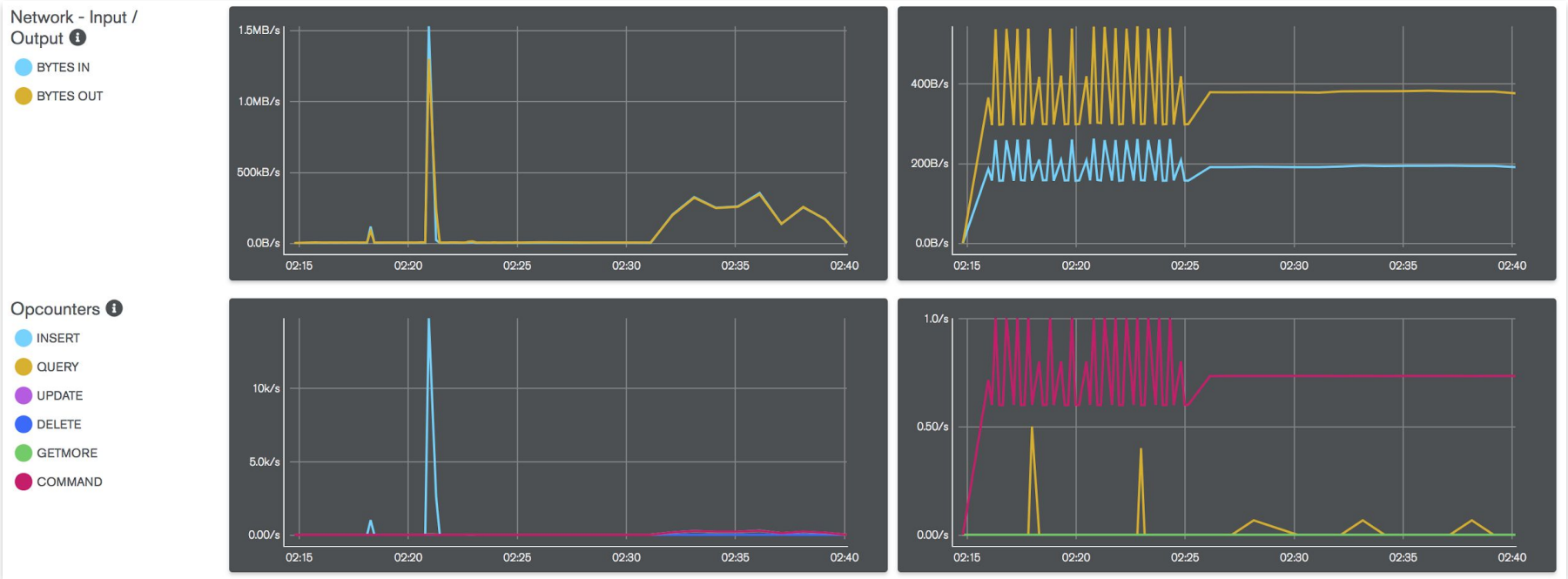
monitoring

– use mongoDB free cloud monitoring: `db.enableFreeMonitoring()`



monitoring

– use mongoDB free cloud monitoring: `db.enableFreeMonitoring()`



challenges

- docker limitations
  - host-container connection under macOS is problematic
- mongodb limitations
  - no support for complex fragmentation
  - aggregation cannot output to sharded collection
- complex queries and aggregation are challenging at first
- setting up HDFS is non-trivial (even using Docker!)

future work

- streamlined multi-node setup (Docker Swarm)
- optimize aggregations and indices
- add authentication (web API, HDFS, mongoDB)
- add simple example UI

Thank you!