

Building high performance event aggregation engine with MongoDB

Adrian Wolny

Krakow MongoDB User Group

a.wolny@avsystem.com

October 29, 2013

- Statement of a Problem
- Solutions
- Demo Application
- Improvements

Problem

Problem Definition and Requirements

- Real-time statistical analytics of numerical data from multiple sources
- Statistics for a given time range must be computed as fast as possible
- Results available with various levels of granularity based on time

In particular

- We define resources to be monitored e.g. session time, upstream/downstream bandwidth usage, stock prices, other time series
- For a given resource we need to collect huge amount of real-time events and provide simple statistics (average, standard deviation, count)
- These statistics must be computed almost instantly (from milliseconds up to a second) and must be available by minute, hour, day, etc.

Solution 1: Keep events in single collection

```
{  
  _id : "wimax/snr/1382925600000",  
  resource: "wimax/snr",  
  time : ISODate("2013-10-28T02:31:13Z"),  
  owner: "00130077e166",  
  value: -65.8  
}
```

Solution 1: Use Aggregation Framework

```
db.events.aggregate({'$match': {'resource' : 'wimax/snr'}},
  {'$project': {
    'value': 1,
    'sqrValue': {'$multiply': ['$value', '$value']},
    'date': {
      'y': {'$year': '$time'},
      'm': {'$month': '$time'},
      'd': {'$dayOfMonth': '$time'}}}},
  {'$group': {
    '_id': {
      'y': '$date.y',
      'm': '$date.m',
      'd': '$date.d' },
    'count': {'$sum': 1 },
    'sum': {'$sum': '$value'},
    'sqrSum': {'$sum': '$sqrValue'}}})
```

Solution 2: Aggregate data from events

How to aggregate avg and stdDev?

Average is obvious!

Values of two power sums s_1 and s_2 are computed for a set of event values, denoted as x_1, \dots, x_N

$$s_j = \sum_{k=1}^N x_k^j$$

Given the results of these summations, the values N, s_1, s_2 can be used at any time to compute the current value of the standard deviation:

$$\sigma = \sqrt{\frac{Ns_2 - s_1^2}{N}}$$

Key operations: **upsert** and **\$inc**

Solution 2: Sample hourly aggregate

```
{  
  _id : "sampleResource/1382925600000",  
  date : ISODate("2013-10-28T02:00:00Z"),  
  minute : {  
    0 : {  
      count : NumberLong(10),  
      sqrSum : 218839.53606779975,  
      sum : 1385.5528225203282  
    },  
    1 : {  
      count : NumberLong(11),  
      sqrSum : 140587.5665313373,  
      sum : 1152.5066740934776  
    },  
    ...  
  }  
}
```

Demo Application

Download from GitHub and play around

github.com/wolny/mongodb-aggregation

Improvements

- Consider event batching at the application level, update higher level aggregates less often
- Consider documents pre-allocation: prevents document/indexes migrations on disk, no padding = more compact representation
- How to manage data growth?

References

- docs.mongodb.org/manual/
- docs.mongodb.org/ecosystem/use-cases/

The End