Optimierung, komplexe Abfragen und sharding

Änderung Datenmodell

- Collection **`stations`** `(update on copy)`
- Beispiel:

```
{
   „stationID“ : „23“, // Identifier (string)
   „creationTS“ : 1314277800000, // Versionierung
   … // Metadaten
}
```

- Abfrage:

```
db.stations.find({„stationID“ : 23, creationTS : { $lt :
   1314277700000}}).sort({„creationTS“ : -1}).limit(1);
```

Optimierung – document size

- Collection **measurings**
- Beispiel:

```
{
„timestamp“ : 1314277800000,
„value“ : 1337,
„stationID“ : „23“,
„partID“ : „wr“,
„serialNo“ : 42,
„datatype“ : „gain“,
[„opt1“ : „string“,]
[„opt2“ : 0]
}
```

```
> db.measurings.stats()
{
        "ns" : "BIS_mongo_eval.measurings",
        "count" : 6000000,
        "size" : 951562096, // coll_size
        "avgObjSize" : 158.59368266666667,
        "storageSize" : 1078239232, // pre_alloc
        "numExtents" : 23,
        "nindexes" : 4,
        "lastExtentSize" : 186253312,
        "paddingFactor" : 1,
        "flags" : 1,
        "totalIndexSize" : 624744512,
        "indexSizes" : {
                "_id_" : 194678736,
                "timestamp_1" : 150953488,
                "datatype_1" : 128158800,
                "value_1" : 150953488
        },
        "ok" : 1
}
```

Optimierung – document size

- Collection **`measurings`**
- Beispiel:

```
{
„a“ : 1314277800000,
„b“ : 1337,
„c“ : „23“,
„d“ : „wr“,
„e“ : 42,
„f“ : „gain“,
[„g“ : „string“,]
[„h“ : 0]
}
```

```
> db.measurings.stats()
{
        "ns" : "BIS_mongo_eval.measurings",
        "count" : 6000000,
        "size" : 647737272, // coll_size
        "avgObjSize" : 107.956212,
        "storageSize" : 891985920, // pre_alloc
        "numExtents" : 22,
        "nindexes" : 4,
        "lastExtentSize" : 155209728,
        "paddingFactor" : 1,
        "flags" : 1,
        "totalIndexSize" : 624744512,
        "indexSizes" : {
                "_id_" : 194678736,
                "a_1" : 150953488,
                "f_1" : 128158800,
                "b_1" : 150953488
        },
        "ok" : 1
}
```

Optimierung – document size

- Collection **measurings_<stationID>**
- Beispiel:

```
{
„a“ : 1314277800000,
„b“ : 1337,
„d“ : „wr“,
„e“ : 42,
„f“ : „gain“,
[„g“ : „string“,]
[„h“ : 0]
}
```

```
> db.measurings.stats()
{
        "ns" : "BIS_mongo_eval.measurings",
        "count" : 6000000,
        "size" : 509099848,
        "avgObjSize" : 84.84997466666667,
        "storageSize" : 607436800,
        "numExtents" : 20,
        "nindexes" : 4,
        "lastExtentSize" : 107782144,
        "paddingFactor" : 1,
        "flags" : 1,
        "totalIndexSize" : 624744512,
        "indexSizes" : {
                "_id_" : 194678736,
                "a_1" : 150953488,
                "f_1" : 128158800,
                "b_1" : 150953488
        },
        "ok" : 1
}
```

Optimierung – document size

- Collection **measurings_<stationID>**
- Beispiel:

```
{
„a“ : 1314277800000,
„b“ : 1337,
„e“ : 42,
„f“ : „gain“,
[„g“ : „string“,]
[„h“ : 0]
}
```

```
> db.measurings.stats()
{
        "ns" : "BIS_mongo_eval.measurings",
        "count" : 6000000,
        "size" : 455737340,
        "avgObjSize" : 75.95622333333333,
        "storageSize" : 651083776,
        "numExtents" : 24,
        "nindexes" : 4,
        "lastExtentSize" : 111796224,
        "paddingFactor" : 1,
        "flags" : 1,
        "totalIndexSize" : 624744512,
        "indexSizes" : {
                "_id_" : 194678736,
                "a_1" : 150953488,
                "f_1" : 128158800,
                "b_1" : 150953488
        },
        "ok" : 1
}
```

Komplexe Abfragen 1

- Wieviele Einträge hat Zeitreihe XY insgesamt/im Zeitintervall [von,bis]?

```
db.measurings.find(
{
   "datatype" : <type>,
   "stationID" : <stationID>,
   "serialNo" : <serialNo>,
   "timestamp" :
   {
       $gt : <from>,
       $lt : <to>
   }
}).count();
```

```
>db.measurings.find({ "datatype" : "gain", "stationID" :
   "wendlinghausen2", "serialNo" : 1, "timestamp" : { $gt :
   1269953100000, $lt : 1269970200000 }}).count();
```

Komplexe Abfragen 2

- Wie ist der Wert der Zeitreihe XY zum Zeitpunkt Z?

```
db.measurings.find(
{
   "datatype" : <type>,
   "stationID" : <stationID>,
   "serialNo" : <serialNo>,
   "timestamp" : <t>
},
{
   <projektion> : 1
});
```

```
> db.measurings.find({ "datatype" : "gain", "stationID" :
   "wendlinghausen2", "serialNo" : 1, "timestamp" :
   1269953100000 }, { "value" : 1 } );
```

Komplexe Abfragen 3

- Wie sind die Werte der Zeitreihe XY im Zeitintervall [von,bis]?

```
db.measurings.find(
{
    "datatype" : <type>,
    "stationID" : <stationID>,
    "serialNo" : <serialNo>,
    "timestamp" :
    {
          $gt : <from>,
          $lt : <to>
    }
},
{
    <projektion> : 1
});

>db.measurings.find({ "datatype" : "gain", "stationID" : "wendlinghausen2",
    "serialNo" : 1, "timestamp" : { $gt : 1269953100000, $lt : 1269970200000 }}, {
    "value" : 1 } );
```

Komplexe Abfragen 4

- Wie ist der Zeitpunkt des ältesten/neuesten Eintrags in Zeitreihe XY?

```
db.measurings.find(
{
   "datatype" : <type>,
   "stationID" : <stationID>,
   "serialNo" : <serialNo>
},
{
   <projektion> : 1
}).sort(
{
   <sort_attr> : [1|-1] // should be indexed
}).limit(1);

>db.measurings.find({ "datatype" : "gain", "stationID" :
   "wendlinghausen2", "serialNo" : 1}, {"timestamp" :
   1}).sort({"timestamp" : -1}).limit(1); // max
```

Komplexe Abfragen 5 / 1

- Wie ist der maximale/minimale/durchschnittliche Wert der Zeitreihe XY im Zeitintervall [von,bis]?

```
map = function() {
   emit(this.stationID, // group by
   {
        total:this.value,
        count:1,
        avg:0,
        min:this.value,
        max:this.value
   });
}
```

Komplexe Abfragen 5 / 2

```
reduce = function(key, values) {
    var r = {total:0, count:0, avg:0, min:0, max:0};
    if(values.length > 0) {
          r.min = values[0].min;
          r.max = values[0].max;
    }
    values.forEach(function(v) {
          r.total += v.total;
          r.count += v.count;

          if(v.min < r.min) {
                r.min = v.min;
          }
          if(v.max > r.max) {
                r.max = v.max;
          }
    });
    return r;
}
```

Achtung: for all k,vals : reduce(k, [reduce(k,vals)]) == reduce(k,vals)

Komplexe Abfragen 5 / 3

```
finalize = function(k, r) {
   if(r.count > 0)
        r.avg = r.total / r.count;
   return r;
}

db.runCommand(
{
   mapreduce : "measurings",
   map : map,
   reduce : reduce,
   out : { inline : 1 },
   query : { "datatype" : "gain", "stationID" : "wendlinghausen2",
   "serialNo" : 1, "timestamp" : { $gt : 1269953100000, $lt :
   1269970200000 } },
   finalize: finalize
});
```

Komplexe Abfragen 5 / 4

```
{
        "results" : [
         {
                  "_id" : "wendlinghausen2",
                  "value" : {
                           "total" : 812939,
                           "count" : 14,
                           "avg" : 58067.07142857143,
                           "min" : 46287,
                           "max" : 63388
                  }
         }],
         "timeMillis" : 34,
         "counts" :
         {
                  "input" : 14,
                  "emit" : 14,
                  "reduce" : 1,
                  "output" : 1
         },
         "ok" : 1
}
```

Komplexe Abfragen 6

- Wie ist der Verlauf des Wirkungsgrades für den Wechselrichter XY im Zeitintervall [von,bis]?

```
db.runCommand(
{
mapreduce : "measurings",
map : function() {
             var r = {total_pac:0, total_pdc:0, efficiency:0};

             if(this.datatype == "pac") {
                          r.total_pac = this.value;
             } else {
                          r.total_pdc = this.value;
             }
             emit(this.stationID, r);
},
reduce : function(key, values) {
             var r = { total_pac:0, total_pdc:0, efficiency:0 };

             values.forEach(function(v) {
                          r.total_pac += v.total_pac;
                          r.total_pdc += v.total_pdc;
             });

             return r;
},
out : { inline : 1 },
query : { $or : [ { "datatype" : "pac" }, { "datatype" : "pdc" } ], "stationID" : "wendlinghausen2", "serialNo" :
     1, "timestamp" : { $gt : 1269953100000, $lt : 1269970200000 } },
finalize: function(k, r) {
             if(r.total_pdc > 0)
                          r.efficiency = r.total_pac / r.total_pdc;
             return r;
}
});
```

Komplexe Abfragen 7 / 1

- An welchen Tagen hat Zeitreihe XY den Schwellenwert Z über-/unterschritten?

```
db.runCommand(
{
mapreduce : "measurings",
map : function() {
		var r = {date:'', count:1, total:this.value, avg:0, timestamp:this.timestamp}
		// group documents by day
		var day = Math.floor(this.timestamp / 1000 / 60 / 60 / 24);
		emit(day, r);
},
reduce : function(key, values) {
		var r = {date:'', count:0, total:0, avg:0, timestamp:0}

		values.forEach(function(v) {
				r.total += v.total;
				r.count += v.count;
				r.timestamp = v.timestamp;
		});
		return r;
},
out : { replace : "temp" },
query : { "datatype" : "gain" , "stationID" : "wendlinghausen2", "serialNo" : 1 },
finalize: function(k, r) {
		var date = new Date(r.timestamp);
		r.date = date.getDate() + "." + (date.getMonth() + 1) + "." + date.getFullYear();
		if(r.count > 0) {
				r.avg = r.total / r.count;
		}
		return r;
}
});
```

Komplexe Abfragen 7 / 2

```
// query the aggregated data

db.temp.find({"value.total" : { $lt : 100 } }, { "value.date" : 1 });
```

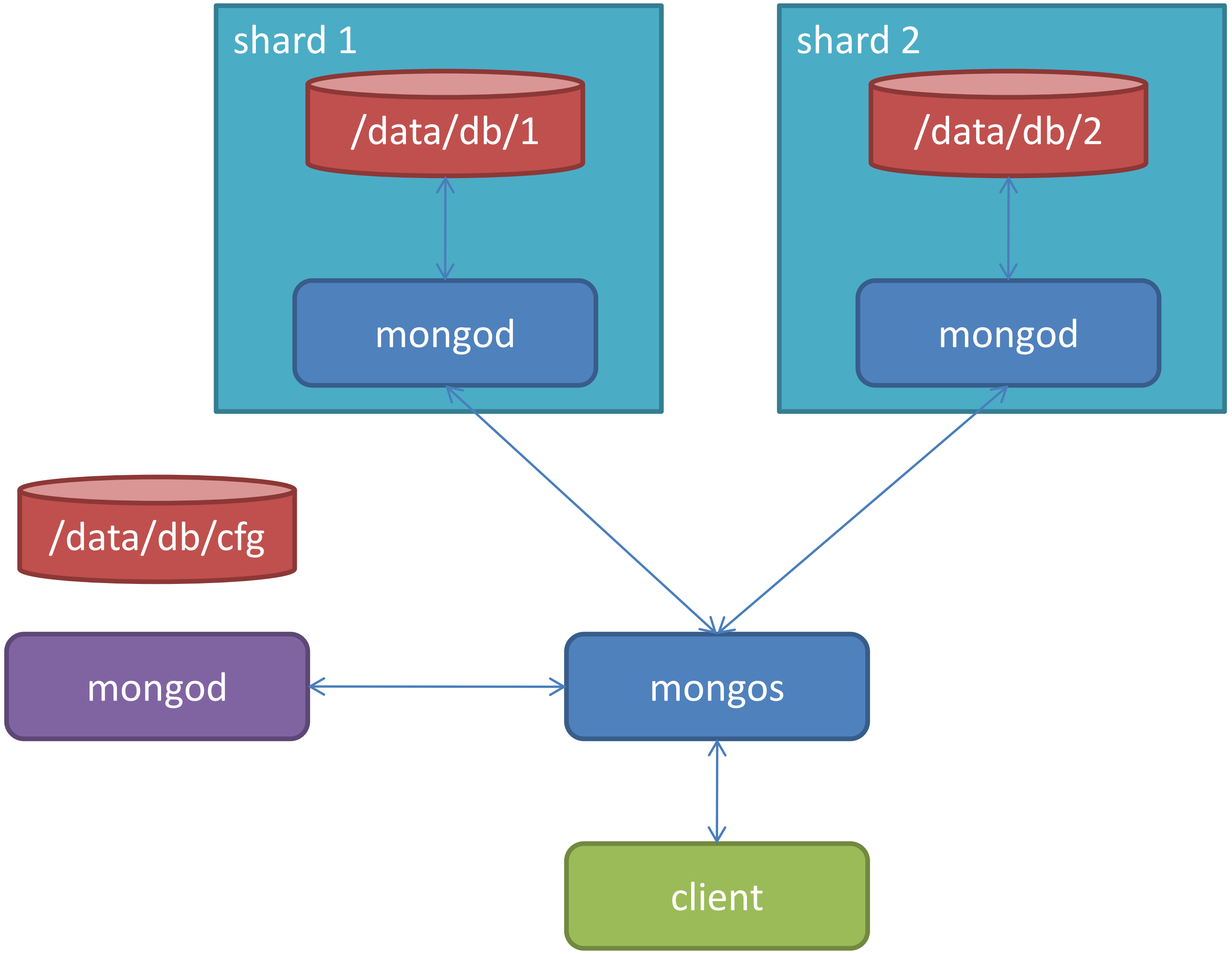
Komplexe Abfragen 8

- Wie groß ist die erzeugte Leistung von Wechselrichter X durchschnittlich pro Temperaturstufe? (d.h. Durchschnitt von PAC je Wert der Temperaturzeitreihe)

Komplexe Abfragen 9

- Für welche Tage im Zeitintervall [von, bis] liegen KEINE Werte für Zeitreihe XY vor?

sharding - Architektur



sharding – server setup

1 the shards

```
> mongod --dbpath=../data/db/1 --port 27017
> mongod --dbpath=../data/db/2 --port 27018
```

2 the configsrv

```
> mongod --configsvr --dbpath=../data/db/config --port 27019
```

3 mongos

```
> mongos --configdb localhost:27019 --port 27020
```

sharding – sharding setup 1

```
> mongo localhost:27020/admin

mongos> db.runCommand({addshard : "localhost:27017" });
{ "shardAdded" : "shard0000", "ok" : 1 }

mongos> db.runCommand({addshard : "localhost:27018" });
{ "shardAdded" : "shard0001", "ok" : 1 }

mongos> db.runCommand( { listshards : 1 } );
{
        "shards" : [
                {
                        "_id" : "shard0000",
                        "host" : "localhost:27017"
                },
                {
                        "_id" : "shard0001",
                        "host" : "localhost:27018"
                }
        ],
        "ok" : 1
}

mongos> db.runCommand( { enablesharding : "BIS_mongo_eval" } );
{ "ok" : 1 }

mongos> db.runCommand({shardcollection : "BIS_mongo_eval.measurings", key : { timestamp : 1 }});
{ "collectionsharded" : "BIS_mongo_eval.measurings", "ok" : 1 }
```

sharding – sharding setup 2

```
mongos> db.printShardingStatus({verbose : 1})
--- Sharding Status ---
  sharding version: { "_id" : 1, "version" : 3 }
  shards:
        {  "_id" : "shard0000",  "host" : "localhost:27017" }
        {  "_id" : "shard0001",  "host" : "localhost:27018" }
  databases:
        {  "_id" : "admin",  "partitioned" : false,  "primary" : "config" }
        {  "_id" : "BIS_mongo_eval",  "partitioned" : true,  "primary" : "shard0000" }
                BIS_mongo_eval.measurings chunks:
                                shard0001       10
                                shard0000       10
                        { "timestamp" : { $minKey : 1 } } -->> { "timestamp" : NumberLong("1269953100000") } on : shard0001 { "t" : 8000, "i" : 1 }
                        { "timestamp" : NumberLong("1269953100000") } -->> { "timestamp" : NumberLong("1303706700000") } on : shard0001 { "t" : 7000, "i" : 2 }
                        { "timestamp" : NumberLong("1303706700000") } -->> { "timestamp" : NumberLong("1303927200000") } on : shard0001 { "t" : 9000, "i" : 2 }
                        { "timestamp" : NumberLong("1303927200000") } -->> { "timestamp" : NumberLong("1305022500000") } on : shard0001 { "t" : 9000, "i" : 3 }
                        { "timestamp" : NumberLong("1305022500000") } -->> { "timestamp" : NumberLong("1305873900000") } on : shard0001 { "t" : 4000, "i" : 0 }
                        { "timestamp" : NumberLong("1305873900000") } -->> { "timestamp" : NumberLong("1306164600000") } on : shard0001 { "t" : 8000, "i" : 4 }
                        { "timestamp" : NumberLong("1306164600000") } -->> { "timestamp" : NumberLong("1307038500000") } on : shard0001 { "t" : 8000, "i" : 5 }
                        { "timestamp" : NumberLong("1307038500000") } -->> { "timestamp" : NumberLong("1307439000000") } on : shard0001 { "t" : 7000, "i" : 4 }
                        { "timestamp" : NumberLong("1307439000000") } -->> { "timestamp" : NumberLong("1308294000000") } on : shard0001 { "t" : 7000, "i" : 5 }
                        { "timestamp" : NumberLong("1308294000000") } -->> { "timestamp" : NumberLong("1308827820000") } on : shard0001 { "t" : 9000, "i" : 0 }
                        { "timestamp" : NumberLong("1308827820000") } -->> { "timestamp" : NumberLong("1309076100000") } on : shard0000 { "t" : 9000, "i" : 1 }
                        { "timestamp" : NumberLong("1309076100000") } -->> { "timestamp" : NumberLong("1310128260000") } on : shard0000 { "t" : 8000, "i" : 9 }
                        { "timestamp" : NumberLong("1310128260000") } -->> { "timestamp" : NumberLong("1310752800000") } on : shard0000 { "t" : 6000, "i" : 2 }
                        { "timestamp" : NumberLong("1310752800000") } -->> { "timestamp" : NumberLong("1311094800000") } on : shard0000 { "t" : 8000, "i" : 2 }
                        { "timestamp" : NumberLong("1311094800000") } -->> { "timestamp" : NumberLong("1312012800000") } on : shard0000 { "t" : 8000, "i" : 3 }
                        { "timestamp" : NumberLong("1312012800000") } -->> { "timestamp" : NumberLong("1312211700000") } on : shard0000 { "t" : 9000, "i" : 4 }
                        { "timestamp" : NumberLong("1312211700000") } -->> { "timestamp" : NumberLong("1313221500000") } on : shard0000 { "t" : 9000, "i" : 5 }
                        { "timestamp" : NumberLong("1313221500000") } -->> { "timestamp" : NumberLong("1313510400000") } on : shard0000 { "t" : 8000, "i" : 6 }
                        { "timestamp" : NumberLong("1313510400000") } -->> { "timestamp" : NumberLong("1314469800000") } on : shard0000 { "t" : 8000, "i" : 7 }
                        { "timestamp" : NumberLong("1314469800000") } -->> { "timestamp" : { $maxKey : 1 } } on : shard0000 { "t" : 1000, "i" : 4 }
```

todo

- Sort Internals (interne Optimierung)
- PAC / PDC für jeden Timestamp (Verlauf)
- „Date“ in Query 7 aus map und reduce rausnehmen
- Komplexe Anfragen 8 und 9
- MapReduce Chunk Definition / Konfiguration
- Sharding Internals näher erläutern
- 1 Milliarde Zeitreihen (duplizieren?, echte Daten?) -> gemeinsames testbed
- Benchmarking (ebenfalls gemeinsames testbed, sonst keine Aussagen möglich)
  - Map Reduce auf sharded DB / single DB
  - JSON Queries auf sharded DB / single DB
- Schriftliche Ausarbeitung zum Vergleich GraphDB / Document Store
- Abschlusspräsentation vorbereiten