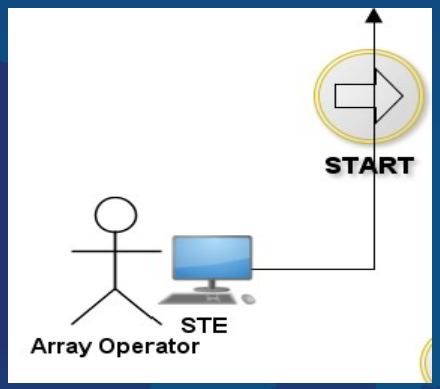
ICALEPS 2013
Exploring No-SQL Alternatives for ALMA Monitoring System

Overview

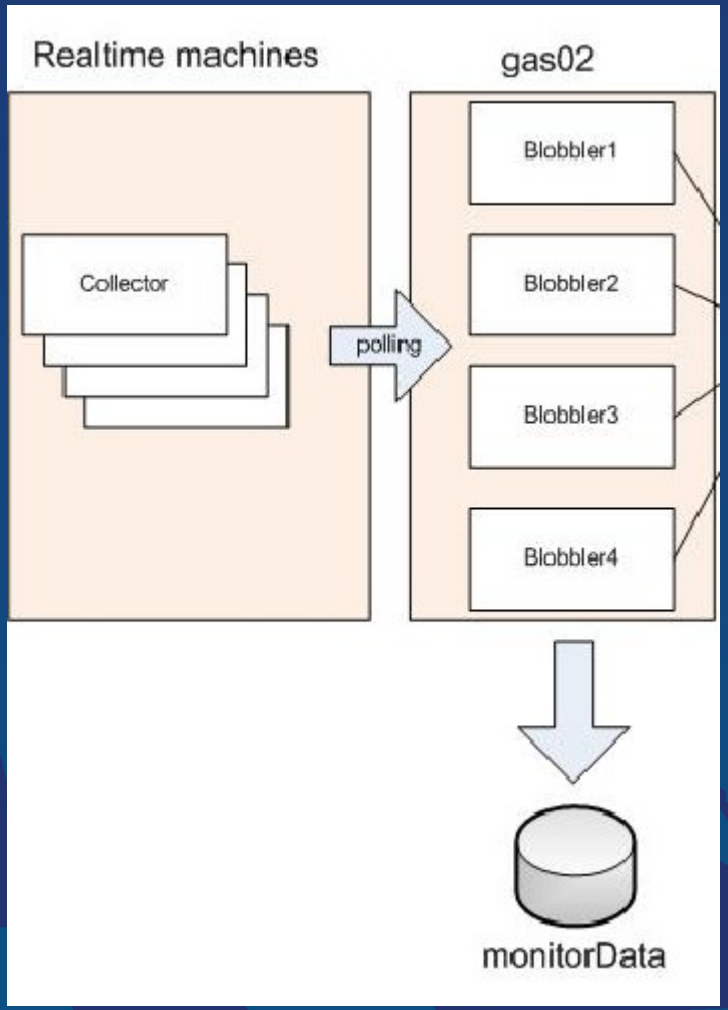
- The current paradigm (CCL and Relational DataBase)
- Propose of a new monitor data system using NoSQL
- Monitoring Storage Requirements
- The Workaround Data Flow
- NoSQL
- Relational DataBase vs NoSQL
- Monitoring System - High Software Level Design
- Monitoring System - Used Software Tools
- RedisIO
- MongoDB
- Monitoring System - Real Time Data in Web Graphics
- Monitoring System - Archive Files in Web Interface
- Monitoring System - Historical Data in Web Interface
- Conclusions

The current paradigm (CCL)

```
pmerino@localhost:~
File  Edit  View  Search  Terminal  Help
--------------------------------------------------------------------------------
                                    _]Q
                                 ,_^_,^ -;
                                        <
                              _c,"      -,
                             ]Q,         <
                              $Qa        -,
                        j.     9QQw,       <
                        4g,     -?$QQwac.   )a
                        -4Qa,      -?9$WQQQQD'
                    y     "$Qga,      .    -
                    4g,     "?$WQmwwwmP
                     ?Qw,       -!??"`
                      -?QQgas_=_a,
                        -??YTTT"

                    Control Command Language
--------------------------------------------------------------------------------
ABM da65-abm: on. DA65   Controller: Degraded. AmbManager: off.
ABM da64-abm: on. DA64   Controller: Degraded. AmbManager: on.
ABM lo-art-1: on. AOSTiming  Controller: Operational. AmbManager: off.
ABM cob-dmc-01:  on. CentralLO Controller (DMC): Operational. AmbManager: off.
--------------------------------------------------------------------------------

. 'cclhelp()' displays the loaded CCL device types, functions and variables
. 'help(object)' provides further information about an object


TFSD [1]>da64_acd=ACD('DA64')

TFSD [2]>da64_acd.GET_STATUS()
TFSD [2]>([131, 64], 135919045413321446L)
```



The current paradigm (Relational DataBase)



Propose of the a new monitor data system using NoSQL

Objectives:

1. Provide Data in Real Time of monitor points of antenna's devices using Web Graphics

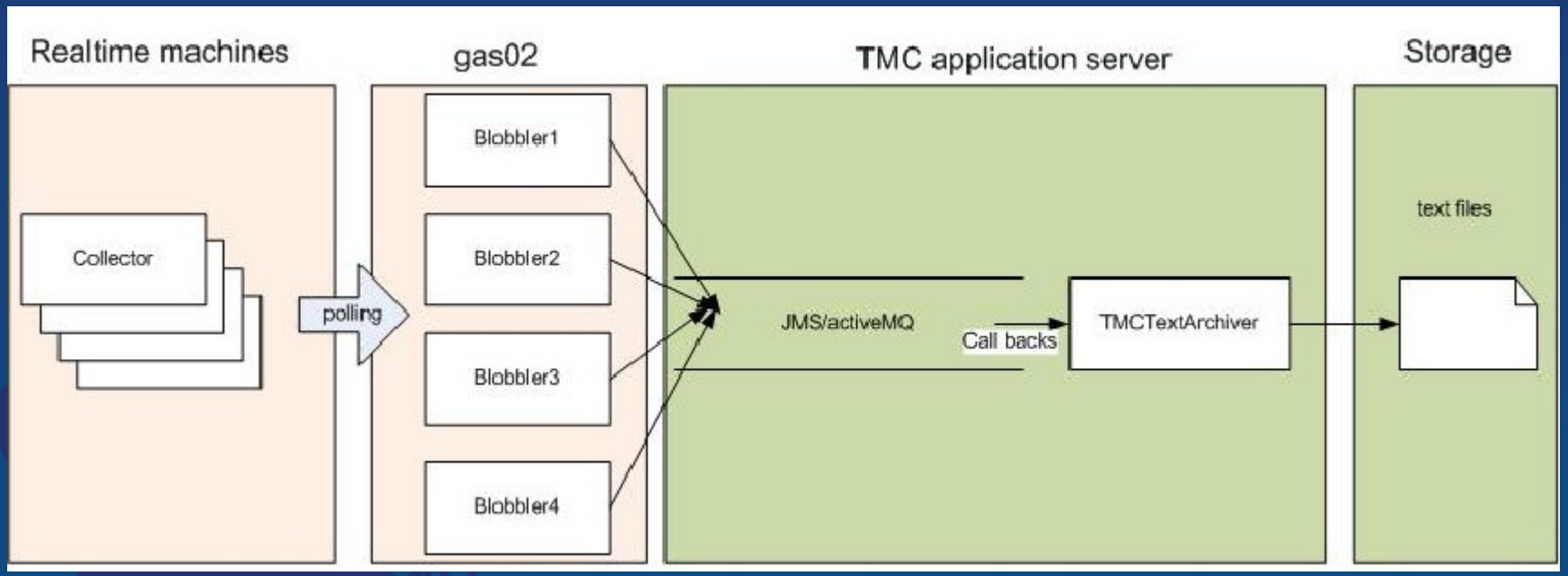
2. Provide Text Files of monitor points of antenna's devices using Web Interface

3. Provide Historical Retrieval Data of monitor points of antenna's devices using Web Interface

Monitoring Storage Requirement

- 66 antennas + CentralLO's + CORR's devices + WeatherStationController + AOSTiming
- Total date rate: ~6.000 – 7.000 clobs/s
  - ~82.9 clobs/s/antenna
- # Monitor Points per antenna type
  - DV/DA: 2.179
  - CM: 2.438
  - PM: 2.474
  - Total monitor points: ~130.000 – 150.000
- Currents Size (MB) of Monitor Data per antenna type (daily)
  - DV: 241
  - DA: 246
  - CM: 301
  - PM: 296
- Current daily monitoring data size: ~25 - 30 GB (in ~120k files)

The Workaround Data Flow



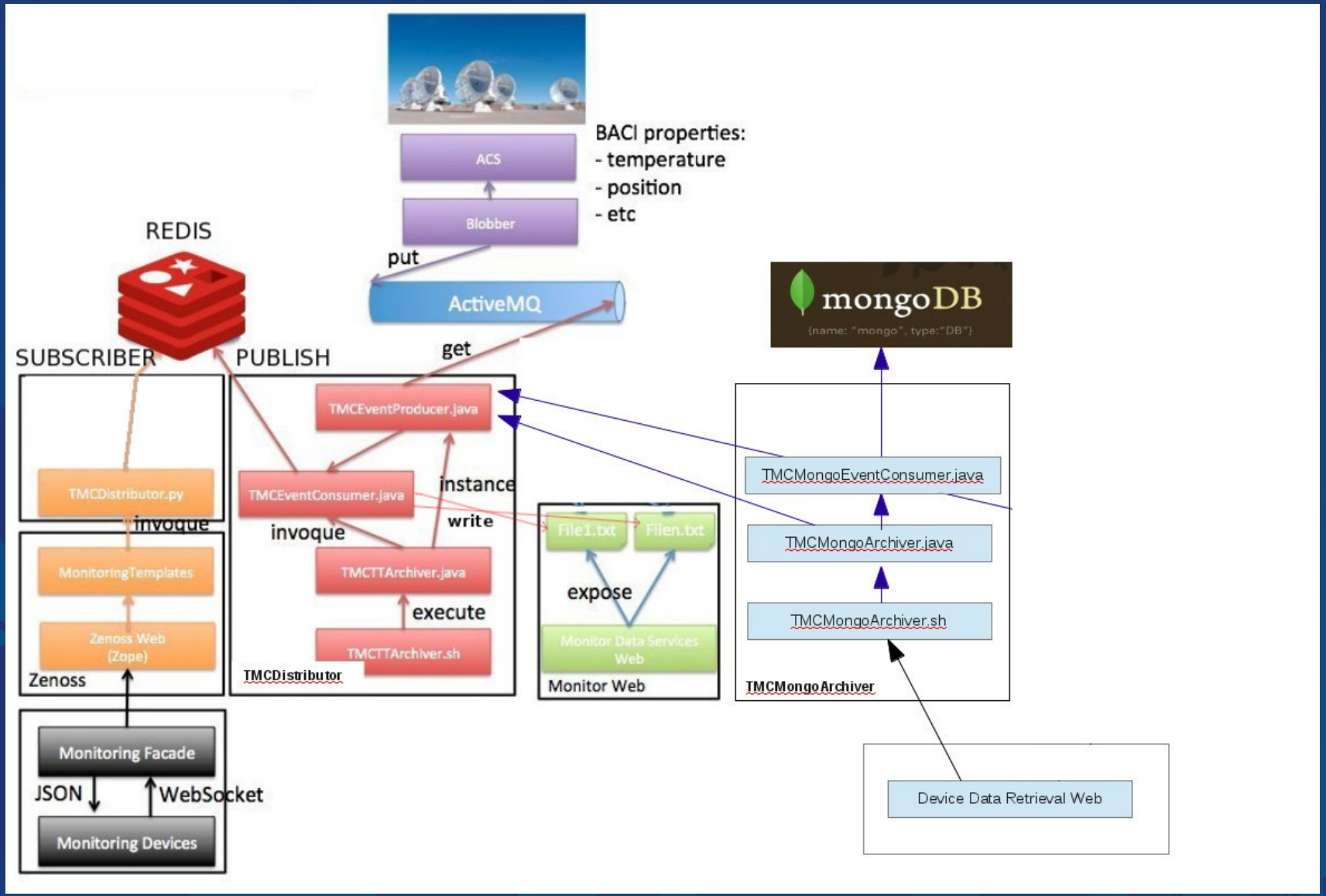
NoSQL

- A NoSQL database provides a mechanism for storage and retrieval of data that uses looser consistency models rather than traditional relational databases.
- Motivations for this approach include simplicity of design, horizontal scaling and finer control over availability.
- NoSQL databases are often highly optimized key–value stores intended for simple retrieval and appending operations, with the goal being significant performance benefits in terms of latency and throughput
- NoSQL database find significant and growing industry use in big data and real-time web applications

Relational DataBase vs NoSQL

Relational DataBase	NoSQL
Relational database management systems are transaction-based and have ACID (Atomicity-Consistency-Isolation-Durability) rules	NoSQL systems do not fully support the ACID (Atomicity-Consistency-Isolation-Durability) rules and there is no transaction concept in many NoSQL systems
Data in the relational database management systems is located on fixed tables and columns	NoSQL systems are not dependent on fixed tables and columns
SQL query is used in Relational database systems	SQL query is not used in NoSQL systems
Disintegration of data by primary key is not compulsory in relational database management systems	NoSQL systems access the data over primary keys

Monitoring System
High Software Level Design



Monitoring System
Used Software Tools

- Apache ActiveMQ (http://activemq.apache.org/)
- Redis IO (http://http://redis.io)
- MongoDB (http://www.mongodb.org)
- Log4j (http://logging.apache.org/log4j/1.2)
- Apache Commons Pool (http://commons.apache.org/pool)
- SpringSource (http://www.springsource.org)
- Junit (https://github.com/kentbeck/junit/wiki)
- Pyramide (http://docs.pylonsproject.org/en/latest/index.html)
- WebSockets (http://www.websocket.org)
- HighCharts (http://www.highcharts.com)

Redis IO

- Redis is an open source, BSD licensed, advanced key-value store. It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets, sorted sets and channels.

- In order to achieve its outstanding performance, Redis works with an in-memory dataset. Depending on your use case, you can persist it either by dumping the dataset to disk every once in a while, or by appending each command to a log.

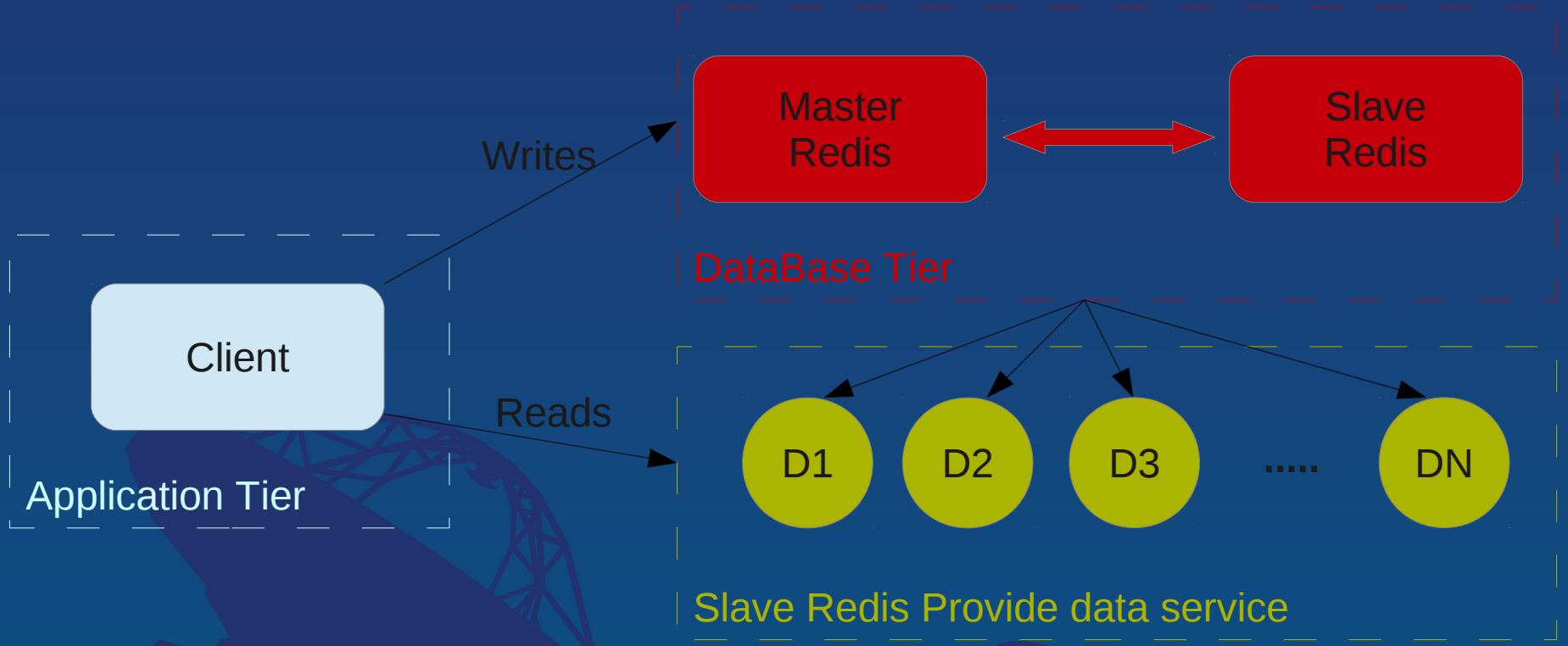
- Redis also supports trivial-to-setup master-slave replication, with very fast non-blocking first synchronization, auto-reconnection on net split and so forth.

- Other features include Transactions, Pub/Sub, Lua scripting, Keys with a limited time-to-live, and configuration settings to make Redis behave like a cache.

- You can use Redis from most programming languages out there.

- Redis is written in ANSI C and works in most POSIX systems like Linux, *BSD, OS X without external dependencies. Linux and OSX are the two operating systems where Redis is developed and more tested, and we recommend using Linux for deploying. Redis may work in Solaris-derived systems like SmartOS, but the support is best effort. There is no official support for Windows builds, but Microsoft develops and maintains a Win32-64 experimental version of Redis.

Redis IO - Scalability



Redis IO - Used Lists

```
redis 127.0.0.1:6379> lrange "TMCS:CONTROL/DV01/LORR:VDC_7" 0 -1
 1) "2013-06-29 13:59:26|13591806649685100;13591806669674383;1372513831093;13591806631093199;0|6.539683\n;6.539683"
 2) "2013-06-29 14:00:30|13591806709621367;13591806729622461;1372513891093;13591806691093199;0|6.5347986\n;6.5347986"
 3) "2013-06-29 14:01:42|13591806769624334;13591806789623169;1372513951093;13591806751093199;0|6.5347986\n;6.5347986"
 4) "2013-06-29 14:02:55|13591806829623166;13591806849627783;1372514011093;13591806811093199;0|6.539683\n;6.539683"
 5) "2013-06-29 14:04:53|13591806889617046;13591806909642974;1372514071093;13591806871093199;0|6.539683\n;6.539683"
 6) "2013-06-29 14:06:27|13591806949623570;13591806969625745;1372514131093;13591806931093199;0|6.539683\n;6.539683"
 7) "2013-06-29 14:07:41|13591807009684158;13591807029695570;1372514191093;13591806991093199;0|6.5347986\n;6.5347986"
 8) "2013-06-29 14:08:55|13591807069621257;13591807089622594;1372514251093;13591807051093199;0|6.5347986\n;6.5347986"
 9) "2013-06-29 14:10:03|13591807129621726;13591807149626826;1372514311093;13591807111093199;0|6.5347986\n;6.5347986"
10) "2013-06-29 14:11:44|13591807189624727;13591807209651985;1372514371093;13591807171093199;0|6.539683\n;6.539683"
11) "2013-06-29 14:12:47|13591807249620423;13591807269631616;1372514431093;13591807231093199;0|6.539683\n;6.539683"
12) "2013-06-29 14:13:57|13591807309619573;13591807329622355;1372514491093;13591807291093199;0|6.539683\n;6.539683"
13) "2013-06-29 14:15:05|13591807369680290;13591807389684125;1372514551093;13591807351093199;0|6.539683\n;6.539683"
14) "2013-06-29 14:16:19|13591807429674386;13591807449627824;1372514611093;13591807411093199;0|6.539683\n;6.539683"
15) "2013-06-29 14:19:40|13591808329672494;13591808349668927;1372515511093;13591808311093199;0|6.539683\n;6.539683"
```

Redis IO - Used Channels

```
redis 127.0.0.1:6379> SUBSCRIBE "TMCS:CONTROL/DV01/LORR:VDC_7"
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "TMCS:CONTROL/DV01/LORR:VDC_7"
3) (integer) 1
1) "message"
2) "TMCS:CONTROL/DV01/LORR:VDC_7"
3) "135918085696197570;135918085896709260;1372515751093;135918085510931990|6.539683\n;6.539683"
1) "message"
2) "TMCS:CONTROL/DV01/LORR:VDC_7"
3) "135918086296865760;135918086496302490;1372515811093;135918086110931990|6.539683\n;6.539683"
1) "message"
2) "TMCS:CONTROL/DV01/LORR:VDC_7"
3) "135918086896265000;135918087096517990;1372515871093;135918086710931990|6.539683\n;6.539683"
1) "message"
2) "TMCS:CONTROL/DV01/LORR:VDC_7"
3) "135918087496196360;135918087696275860;1372515931093;135918087310931990|6.5347986\n;6.5347986"
```

Mongo DB

- MongoDB is a document database that provides high performance, high availability, and easy scalability.

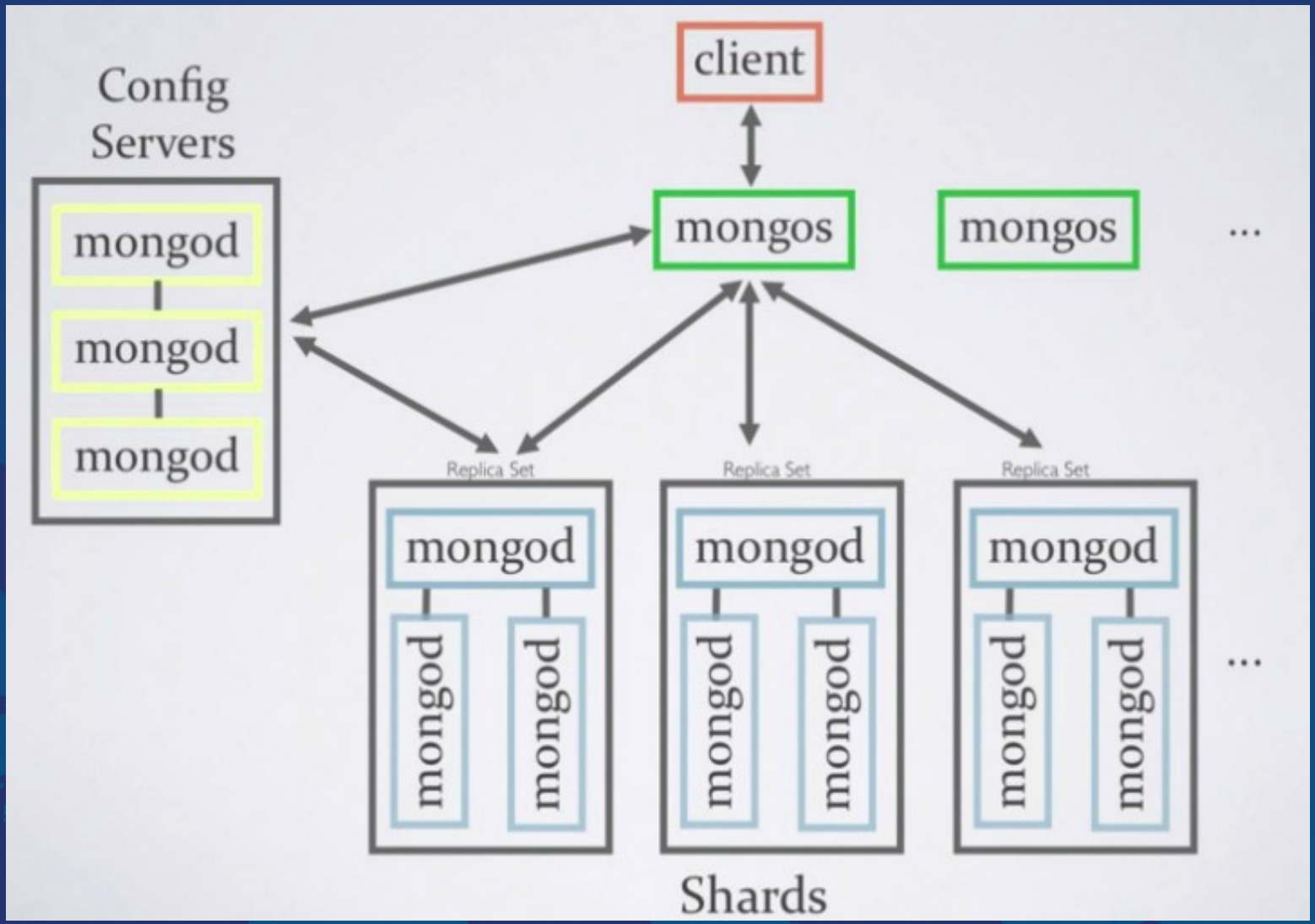
- Document Database
  - Documents (objects) map nicely to programming language data types.
  - Embedded documents and arrays reduce need for joins.
  - Dynamic schema makes polymorphism easier.

- High Performance
  - Embedding makes reads and writes fast.
  - Indexes can include keys from embedded documents and arrays.
  - Optional streaming writes (no acknowledgments).

- High Availability
  - Replicated servers with automatic master failover.

- Easy Scalability
  - Automatic sharding distributes collection data across machines.
  - Eventually-consistent reads can be distributed over replicated servers.

MongoDB - Scalability



MongoDB
One monitor point per document


```
{
    "_id" : { "$oid" : "50520ff925d8b6dfb8b4c353" },
    "date" : { "$date" : 1347554984516 },
    "location" : "TFINT",
    "componentName" : "CONTROL/CM12/DRXBBpr1",
    "propertyName" : "POWER_ALARM_REG_B",
    "monitorPointName" : "POWER_ALARM_REG_B_PSUMMARY",
    "serialNumber" : "10bfae3e010800c9",
    "monitorValue" : "255",
    "acsTime" : 135668477845168070,
    "index" : 0
}
```

MongoDB
A clob per document


```
{
    "_id" : { "$oid" : "50520ff925d8b6dfb8b4c353" }
    "dateStart" : "2012-12-11 11:50:28"
    "dateEnd" : "2012-12-11 11:50:38"
    "location" : "TFINT"
    "componentName" : "CONTROL/DV06/LO2BBpr2"
    "propertyName" : "POWER_SUPPLY_3_VALUE"
    "monitorPointName" : "POWER_SUPPLY_3_VALUE"
    "serialNumber" : "10bfae3e010800c9"
    "clob" : "135745302282272610|-14.713619999999999|135745302382272610|-14.713619999999999"
    "index" : "0"
}
```

MongoDB
A monitor point per day per document

```
{
    _id : "20120901/CM12/DRXBBpr1/POWER_ALARM_REG_B_PSUMMARY",
    metadata : {
        date : "2012-09-01",
        antenna : "CM12",
        component : "DRXBBpr1",
        property : "POWER_ALARM_REG_B",
        monitorPoint : "POWER_ALARM_REG_B_PSUMMARY",
        location : "TFINT",
        serialNumber : "10bfae3e010800c9",
        index : 0,
        sampleTime : 1 },
    hourly : {
        "0" : {
            "0" : {
                "0" : 12345,
                "1" : 12345,
                "2" : 12345,
                ...
                "59" : 12345 },
            "1" : {
                "0" : 12345,
                "1" : 12345,
                "2" : 12345,
                ...
                "59" : 12345 },
            ...
            "59" : {
                "0" : 12345,
                "1" : 12345,
                "2" : 12345,
                ...
                "59" : 12345 } },
        "1" : {
            "0" : {
                "0" : 12345,
                "1" : 12345,
                "2" : 12345,
                ...
                "59" : 12345 },
            "1" : {
                "0" : 12345,
                "1" : 12345,
                "2" : 12345,
                ...
                "59" : 12345 },
            ...
            "59" : {
                "0" : 12345,
                "1" : 12345,
                "2" : 12345,
                ...
                "59" : 12345 } } }
}
```

Monitoring System
Real Time Data in Web Graphics

- Video is Pending

Monitoring System
Archive Files in Web Interface

Monitoring System
Historical Data in Web Interface

- Video is Pending

Conclusions

- NoSQL is a perfect paradigm for store big and heterogeneous data. such as ALMA monitoring data
- RedisIO is an appropriate key-value store for cache storage of ALMA monitoring data
  - Redis Lists are well designed for put/get a lot of values of monitoring data in blocks for future processes
  - Redis Channels are well designed for publishers/subscribers of events in Real Time
- MongoDB is a suitable document oriented alternative for permanent storage of ALMA monitoring data
  - A monitor point per day per document is the best option for extract all needed data in few milliseconds