
allegro group

allegro group

# Product Catalogue – Case Study

**Piotr Modzelewski** **Bartłomiej Rymarski**

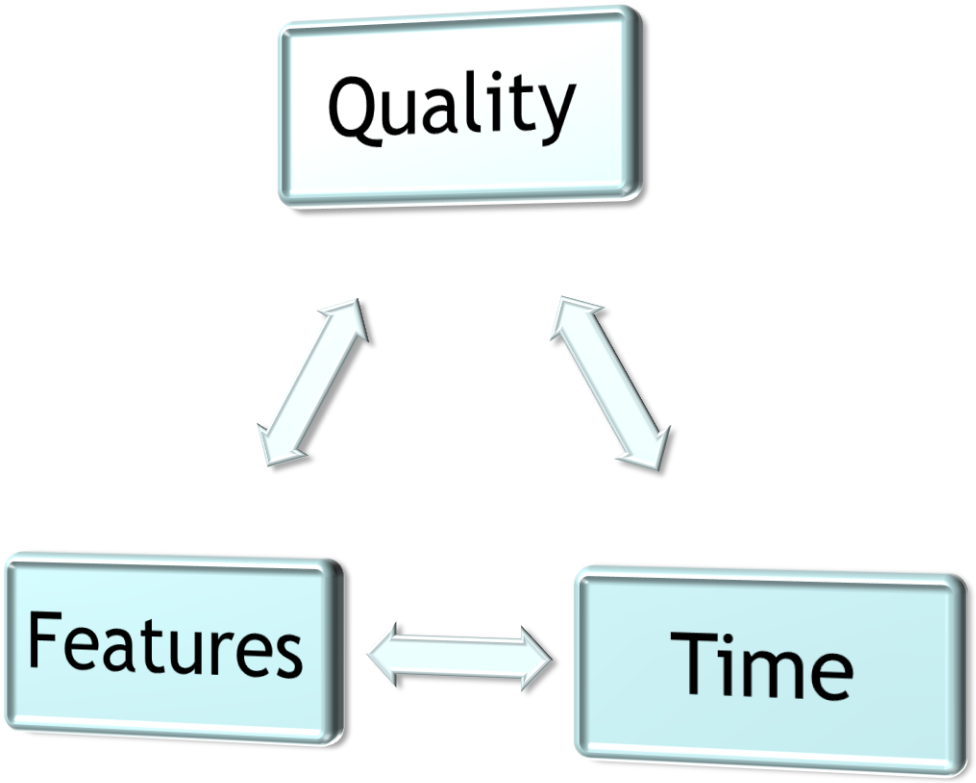
allegro group

# Cheating the project management triangle

an overview of technologies helping to create a scalable and efficient service quickly

allegro group

# Project management triangle with constant budget



allegro group

Apache incubator


Solution for the lazy

allegro group

# WTH?

- Binary protocols are fast
  - Try to make fast server with XML! :)
- Managing binary protocols is hard
  - Is it?
  - Can it be easier?
- Web API should be language independent
  - Binary in PHP?
  - Little Endian vs. Big Endian

allegro group

# So why Thrift?

- Language support:
  - C/C++/D, PHP, Java, Erlang, Perl, Python …
- Why to write it, if you can generate it?
- Stand alone server!
  - No tomcat BS.
- To good to be true?

allegro group

# Implementation Step 1 / 3 - Design

```
namespace java pl.allegro.example.server

typedef i64 Timestamp

service ExampleService {
  Timestamp ping(),
}
```

allegro group

# Implementation Step 2 / 3 – Gen gen

```
thrift --gen java ping.thrift
```

```
<dependency>
        <groupId>slf4j-api</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.5.10</version>
</dependency>
```

```
<dependency>
        <groupId>libthrift</groupId>
        <artifactId>libthrift</artifactId>
        <version>0.2.0</version>
</dependency>
```

allegro group

# Implementation Step 3 / 3 - Coding

```java
package pl.allegro.example.server;

import org.apache.thrift.TException;

public class ExampleServiceImplementation
        implements ExampleService.Iface {

  @Override
  public long ping() throws TException {
    return System.currentTimeMillis();
  }
}
```

```java
package pl.allegro.example.server;

import org.apache.thrift.protocol.*;
import org.apache.thrift.transport.*;
import org.apache.thrift.protocol.TBinaryProtocol.Factory;
import org.apache.thrift.server.TThreadPoolServer;

public class Main {
  public static void main(String[] args) throws Exception {
    TServerSocket serverTransport = new TServerSocket(6667);
    ExampleService.Processor proc = new ExampleService.Processor(
            new ExampleServiceImplementation());
    Factory protFactory = new TBinaryProtocol.Factory(true, true);
    TThreadPoolServer server = new TThreadPoolServer(
            proc, serverTransport, protFactory);
    server.serve();
  }
}
```

allegro group

# Deployment – Step 1/2

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-shade-plugin</artifactId>
   <executions>
        <execution>
                <phase>package</phase>
                <goals> <goal>shade</goal> </goals>
        </execution>
   </executions>
   <configuration> <finalName>
        ${artifactId}-${version}-with-dependencies
   </finalName></configuration>
</plugin>
```

allegro group

# Deployment – Step 2/2

```
> mvn clean install
> java –jar PingService-1.0-with-dependencies
```

allegro group

# So how to solve our problem?

```
namespace java pl.allegro.example.server
typedef i64 Timestamp
struct ApiObject {
    1: i64 id,
    2: string name,
    3: string description,
    4: list<string> images,
}
struct ApiSearchResult {
    1: list<ApiObject> products,
    2: i64 totalHits
}
service PrototypeService {
    Timestamp ping(),
    ApiObject getById (1: i64 id),
    ApiSearchResult search (1: string query, 2: i32 page, 3: i32 limit),
}
```

allegro group

# When DB is not enought.

Cassandra for example?

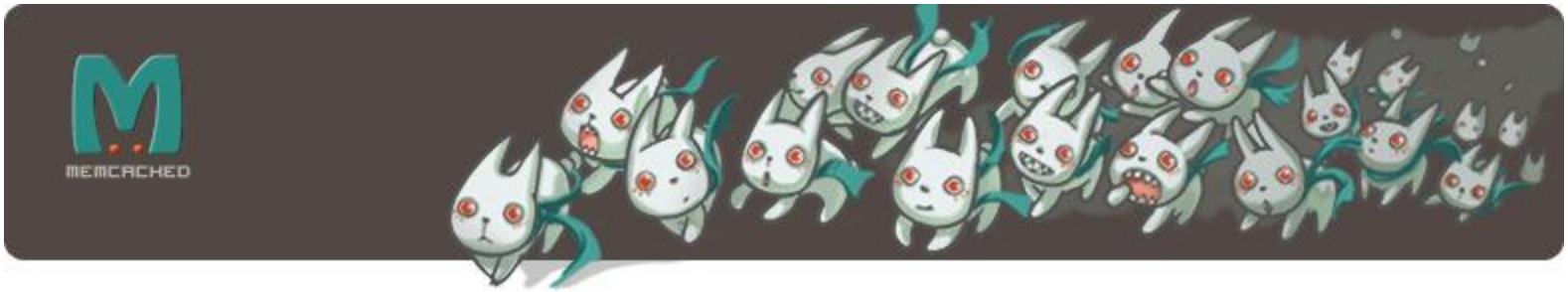
allegro group

# Why Db is not good for our cause?

- Db is heavy
  - Downsides of transactions
  - Select via index vs Memcached
- Db is unpredictable
  - Heavy queries can lock tables
  - Db maintenance queries can become heavy

allegro group

# NoSQL for the rescue!




EHCACHE
a product from TERRACOTTA

redis

Cassandra

allegro group

# Cassandra

- Thrift based Service
- Cluster of agents – Ring
- The data is stored on hard disk and memory (via cache)
- The data distribution is configured
  - It can be requested to have data copied on each server or just on a part of them
- Many options of data storage

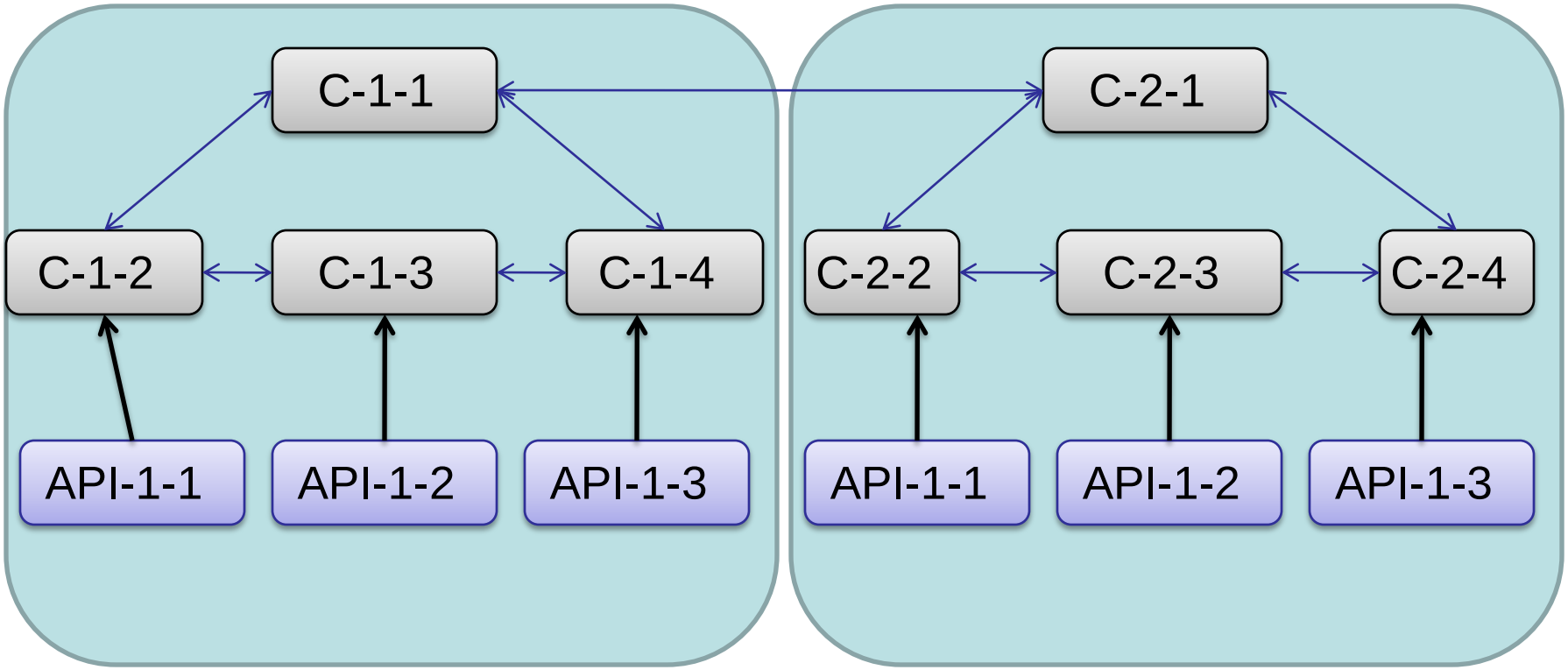
allegro group

# Cassandra in practice

- Keep same data on every node
- Deploy Cassandra near the API server – same machine is the best
- Keep the data in the scheme simple
- The bigger cache, the better
- Help Cassandra to keep things in cache

allegro group

# Example deployment



allegro group

# getById() algorithm

1. Get the byte[] data from the Cassandra by ID
2. If there is no data, throw an exception, that object is not found
3. Unserialize the data as Java object
4. If it isn't Thrift class object, translate the data to a Thrift object
5. Return the object

allegro group


# Solr

Let sphinx be a relic

allegro group

Sphinx

# There was a Sphinx

- Popular and well known search engine
- Speed gained by very simple use only
- Really bad debug support
- Command line tools only
- No good build-in replication method
- Bad support for connection pooling

allegro group

# Solr

- Web based administration panel with statistics, options and query tool
- Extendable by hook-like plugins
- Many build in features
  - Highlights
  - Spellchecking
- Rich schema possibilities

allegro group

# Solr in practice

- It works!
- Master – slave – should be used only if the data modification isn't to often
- Use binary protocol if possible
- Reindex clears cache unless configured otherwise
- Cache can't be bigger then 2GB
- Integrating cache mechanics into Solr costs a lot

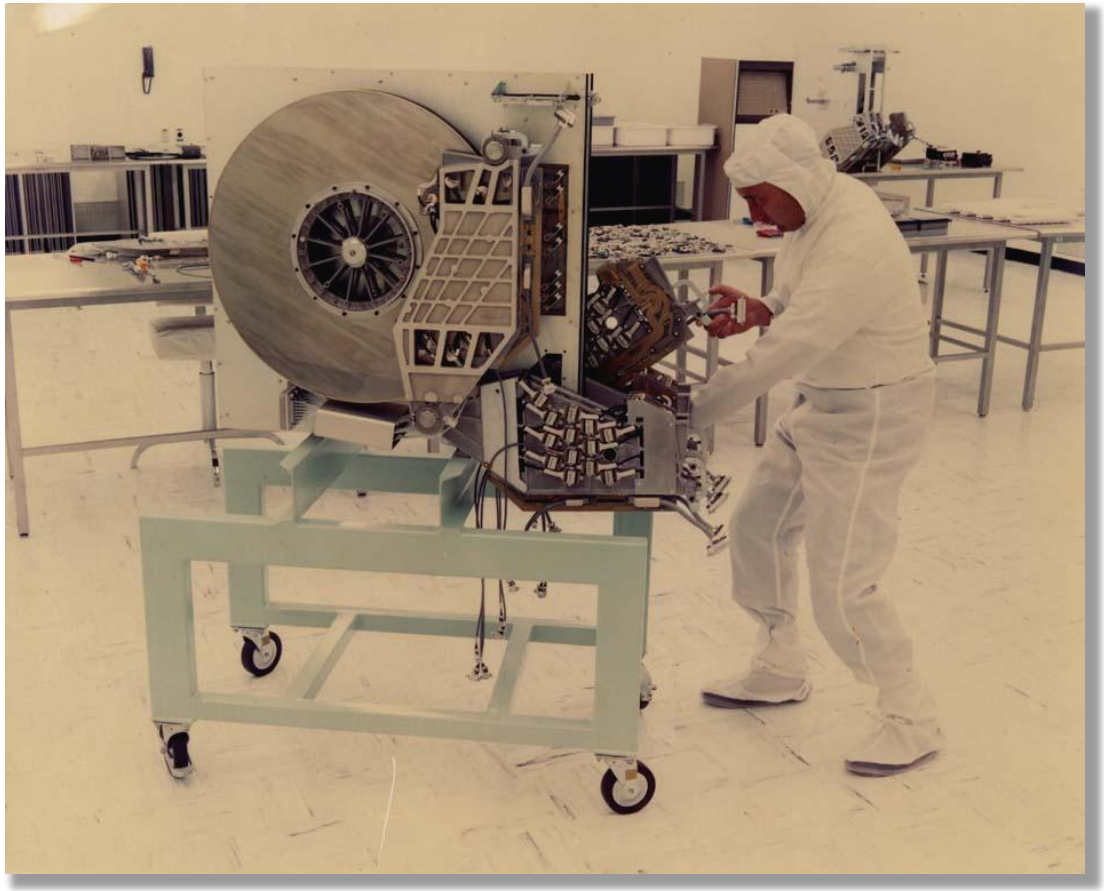
allegro group

# Search() algorithm

1) Process the query
2) Send query to solr
3) Get list of object ids
4) Get objects from cassandra with multiget
5) Unserialize the data as Java object
6) Translate the data to a Thrift object
7) Return the object

allegro group

# MogileFS – OMG files!



allegro group

# What is MogileFS



- Distributed filesystem
- OpenSource
- Developed by Danga Interactive
- Designed for scalable web app storage
- Users:








allegro group

# Why use MogileFS

- Horizontal scalability
- Reliability (no Single Point Of Failure)
- No RAID required (better than RAID)
- Application level (no kernel modules)
- Library available for
  - Perl, PHP, Python, Java, Ruby …

allegro group

# MogileFS Components

- Client
- Tracker
  - Handles communication (Application/Frontend<>Storage/DB)
  - Multiple workers (processes)
- Storage
  - Physical storage for files
  - Any HTTP server (with WebDAV support)
  - Your OS & FS
- Database
  - Stores metadata, server settings, hosts & devices
  - Your RDBMS

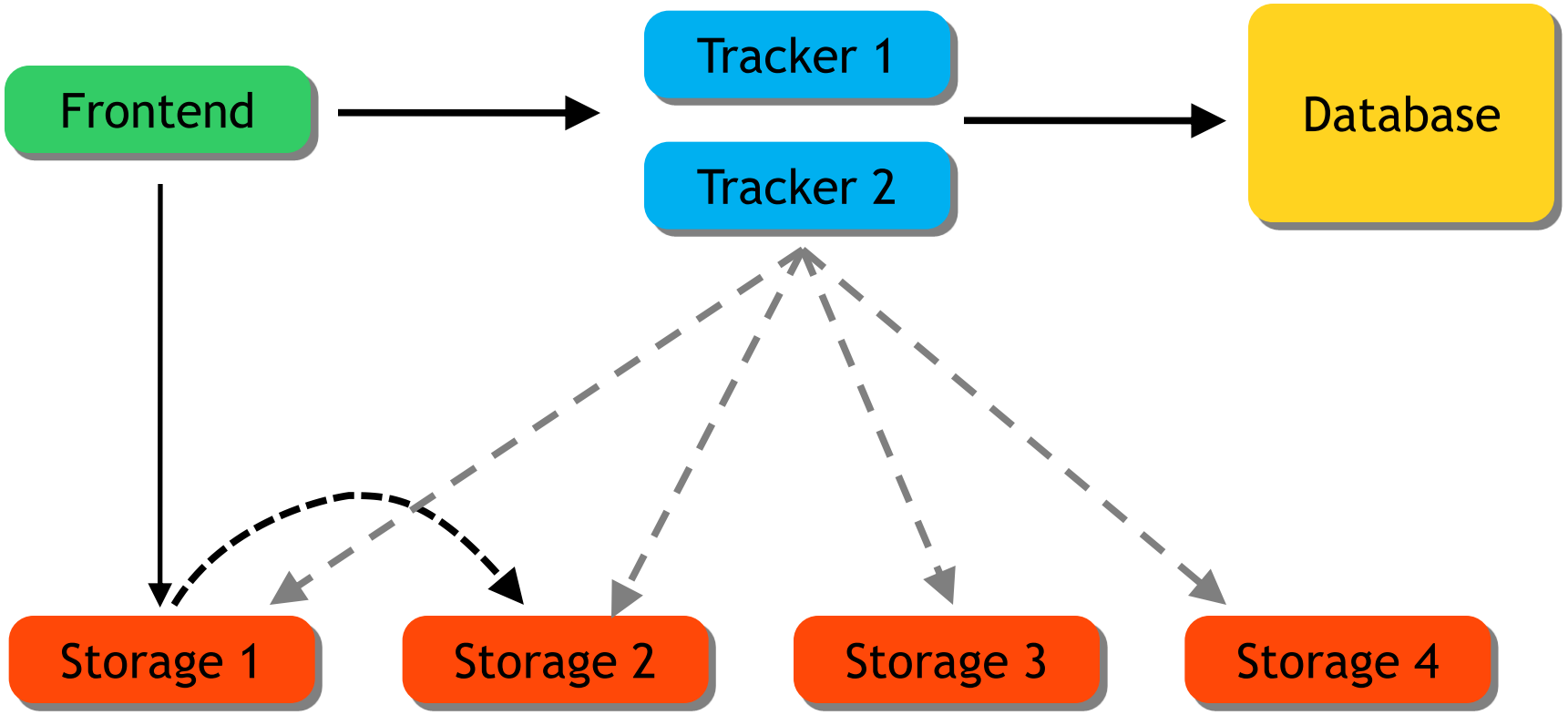
allegro group

# All your files are belong to …

- Files are organized in a flat namespace
  - Classes (replication policy)
  - Domains (many applications)
- Replication according to file class policy
  - Each class has its own minimal replica count
  - You decide how safe your files should be

allegro group

# Overview



allegro group

# Scaling & Performance

- Add more frontends / trackers / workers
- Add more storage nodes / devices
- Increase replicate count
- Add MySQL Slave replica for reads
- Replace Mogstored with Nginx
- Use Nginx as a frontend
- Cache tracker replies (memcached)

allegro group

# SLA

allegro group

# Reliability & HA

- MySQL Master<>Master replication (active/standby)
- Redundant storage nodes / trackers / frontends
- Haproxy for tracker and DB slave connections
- OpenSolaris & ZFS for storage nodes



allegro group

# Multiple datacenters awarness

- MultipleNetworks plugin
  - Directs requests from one IP class to storage nodes on the same address class
  - As easy as

```
# mogadm settings set network_zones dc1,dc2
# mogadm settings set dc1 192.168.0.0/24
# mogadm settings set dc2 10.0.0.0/24
```

- MySQL Master<>Master replication between datacenters

allegro group

# ▶ Usage examples

- Adding a new storage node

```
# mogadm host add storage5 --ip=192.168.0.51 --port=7500 --status=alive
```

- Adding a new device on a storage

```
# mogadm device add storage5 8 --status=alive
```

- Changing replication policy for class1

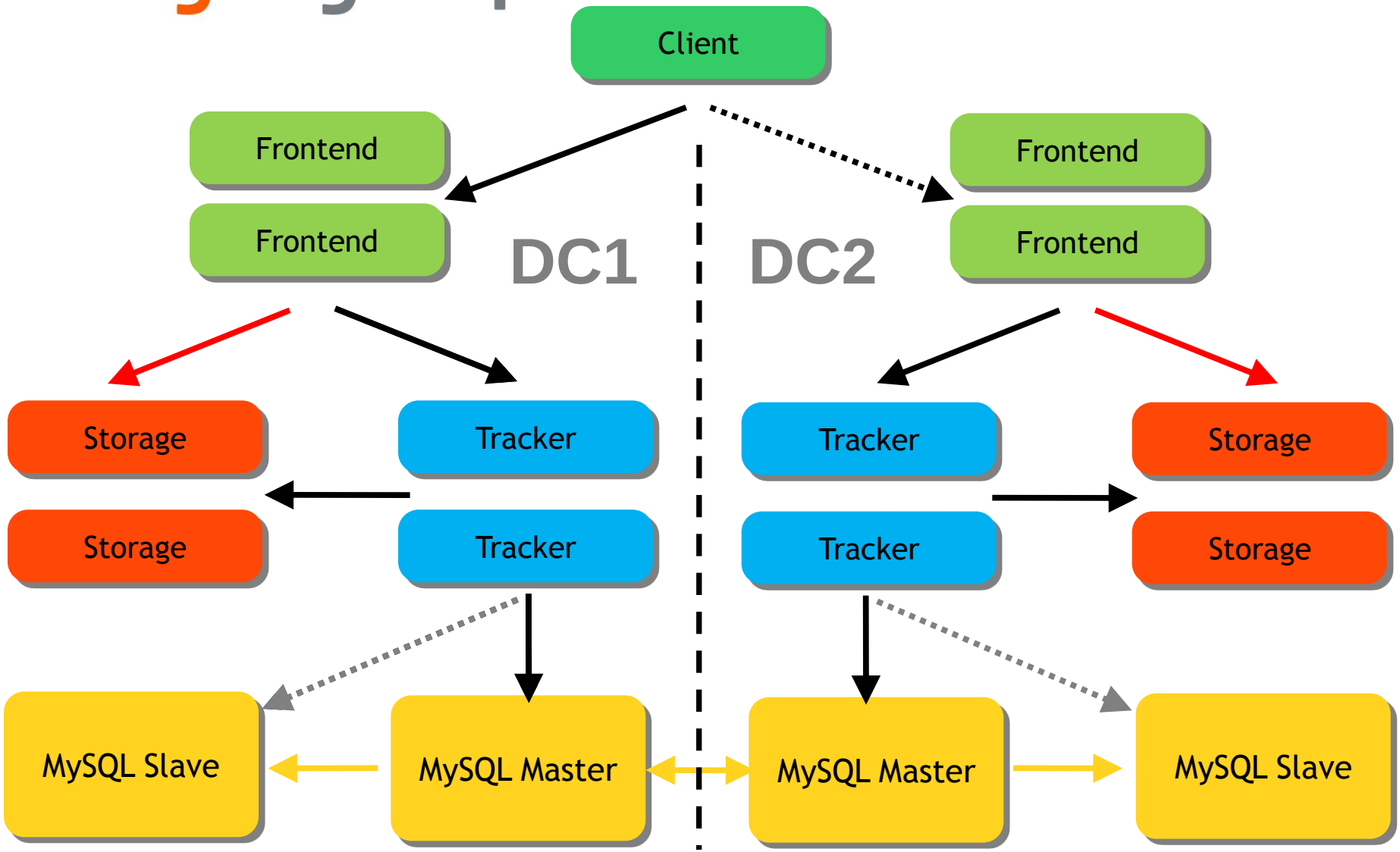
```
# mogadm class modify domainOne class1 --mindevcount=3
```

- Adding a MySQL slave replica for doing reads

```
# mogadm slave add mogdb2 --dsn="DBI:mysql:mogilefs:192.168.0.111"
--username="mogilefs" --password="OMG"
```

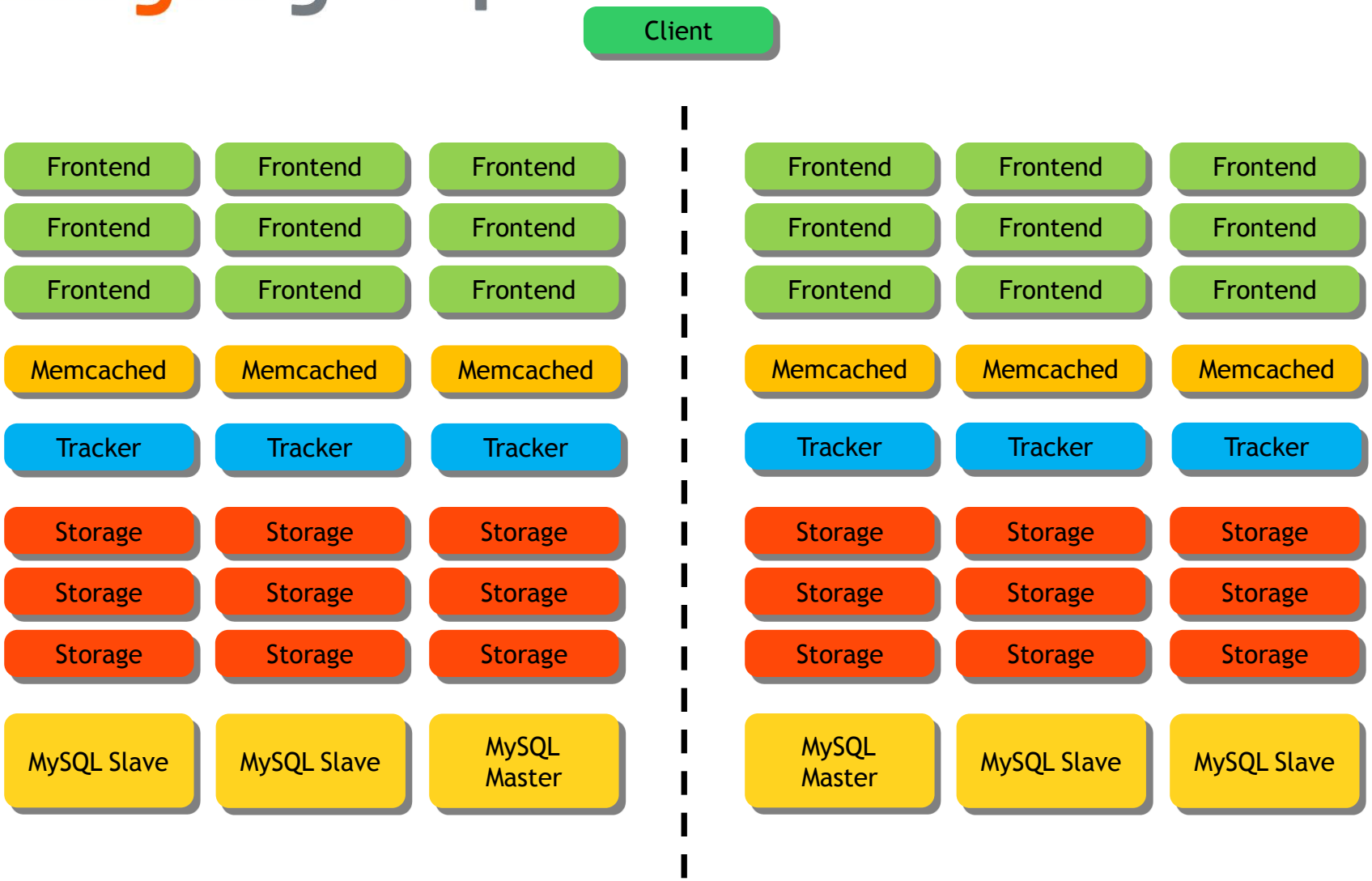
allegro group

# Now



allegro group

# Future



allegro group

Thank you!

Q&A?

**Piotr Modzelewski**
**piotr.modzelewski@allegro.pl**

**Bartłomiej Rymarski**
**bartlomiej.rymarski@allegro.pl**