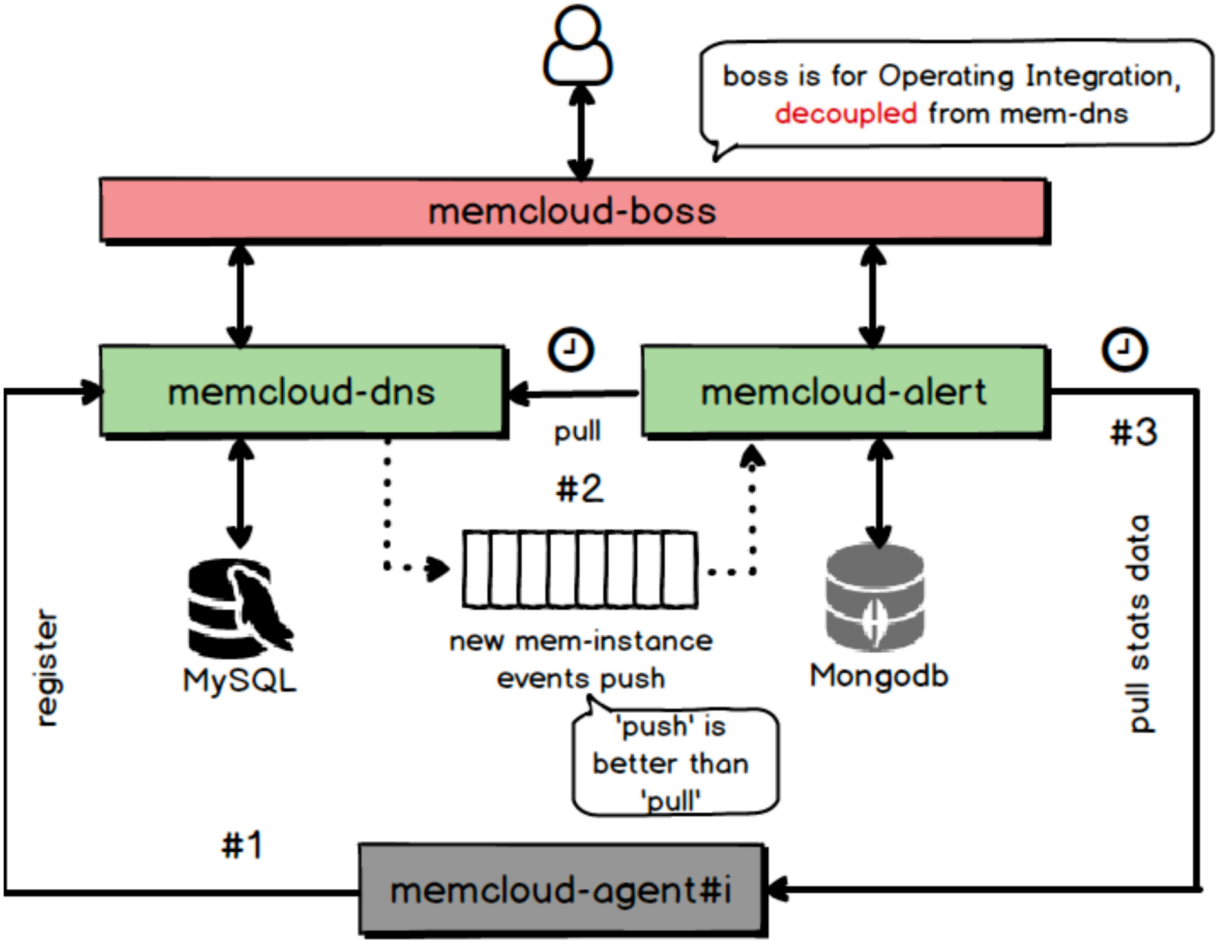
架构设计优化

1 整体建议

现有架构一个设计不合理的地方就是把 `运营后台` （别称 `mem-boss` ）的Web界面功能全部耦合在 `mem-dns` 里。导致 `mem-dns` 系统，既需要依赖MySQL，又需要依赖Mongodb，还需要依赖MemAlert应用。一个更加清晰的架构应该是：

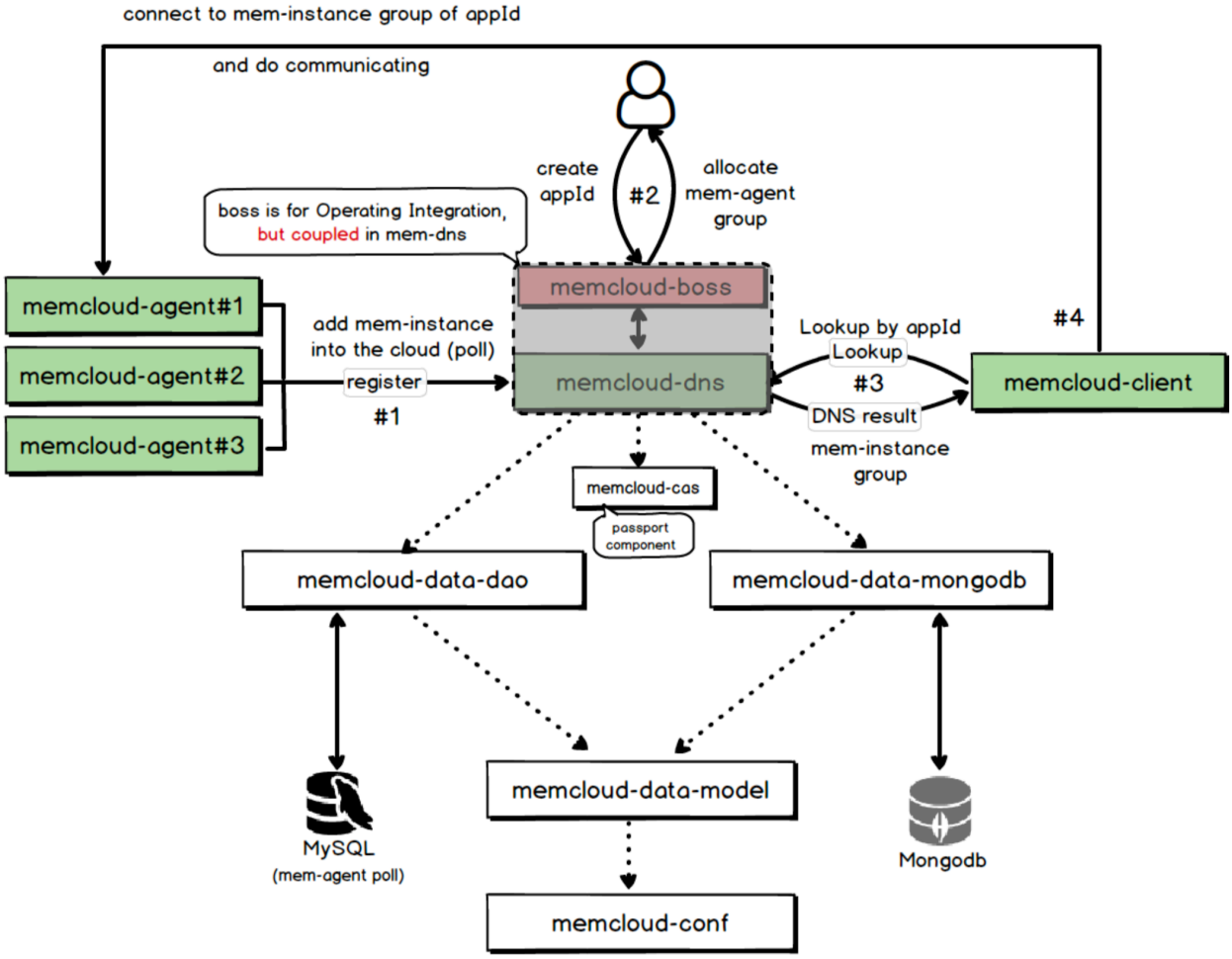


上图清晰展示：

1. **mem-agent注册**： 当我们有一台新机器时，在上面用mem-agent软件包安装 `mem-instance` ，启动，并关联好它的peer，以成对的方式注册到mem-dns上。
2. **mem-alert监控**： 当mem-dns中加入了一个men-agent，则会给推送一条信息，以便men-alert开始对mem-agent发起统计信息的采集。为了弥补消息推送的丢失，mem-alert还可以隔时通过mem-dns查询最近一天内新注册的mem-agent列表，以此进行端到端的对账。
3. **mem-boss运营**： 运营管理人员通过mem-boss进行一站式集中管理。BOSS可以访问mem-dns以查询mem-agent的缓存池，对新注册的App分配mem-agent的集群；还可以访问mem-alert，对每个mem-instance展示访问量变化曲线等。

2 从DNS解耦出BOSS

但是现有的设计，原本属于mem-boss的运营管理功能，全部耦合在mem-dns中。如下图所示：



导致mem-dns既要依赖 `memcloud-data-dao` ，又要依赖 `memcloud-data-mongodb` 。

原本依赖关系应该这样：

- **mem-dns条线**： `mem-dns` --> `memcloud-data-dao` --> MySQL
- **mem-alert条线**： `mem-alert` --> `memcloud-data-mongodb` --> Mongodb

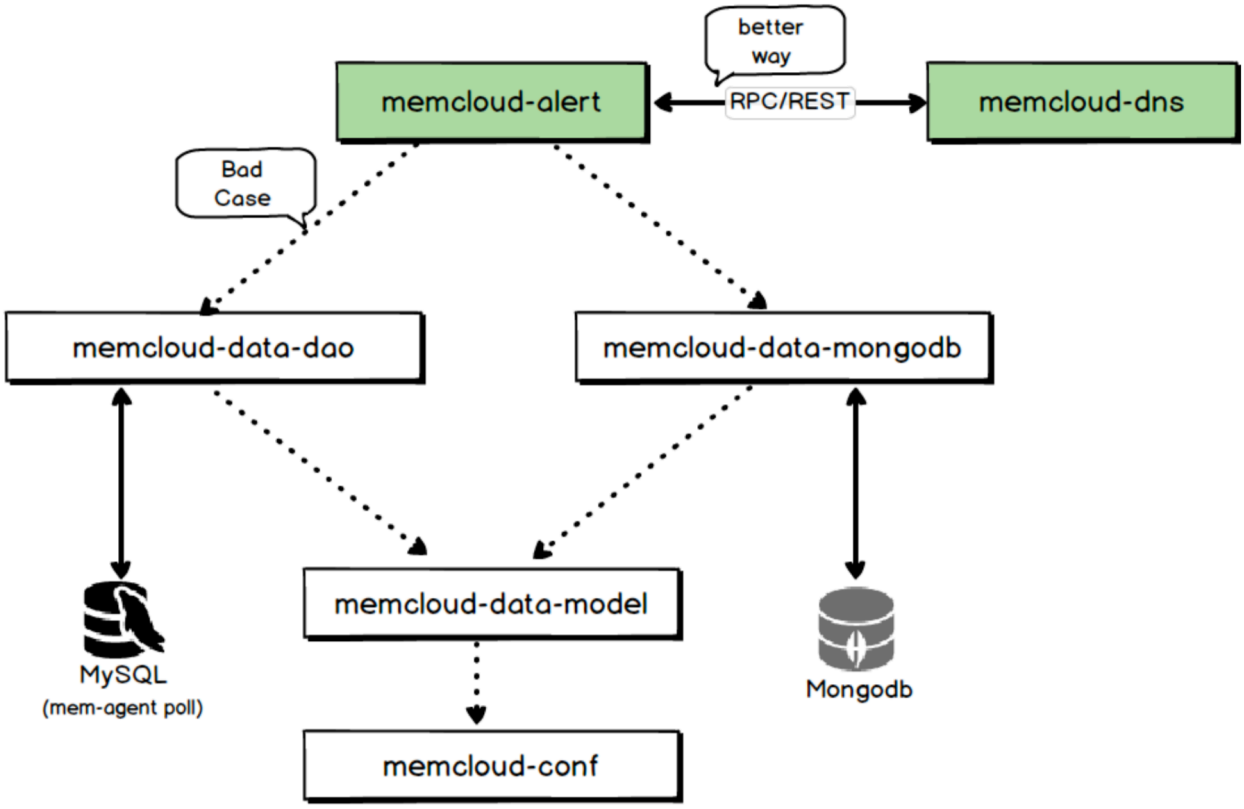
一站式集中管理：

- 左手依赖： `mem-boss` --> `mem-dns`
- 右手依赖： `mem-boss` --> `mem-alert`

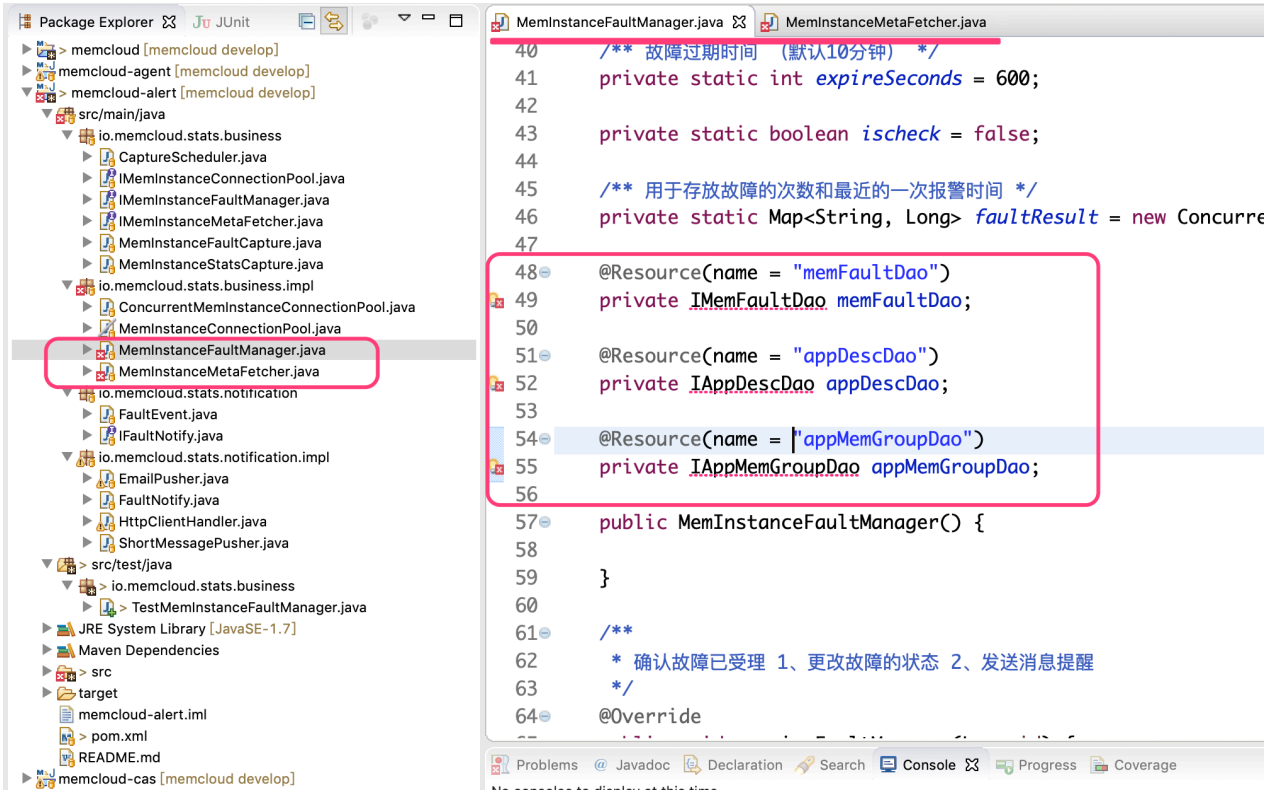
3 从Alert删除DB访问

现在 `mem-alert` 为了获取APP与Memcached实例的对应关系，是直接访问mem-dns侧的数据库的。也就是说， `mem-dns` 与 `mem-alert` 微服务不够彻底，它们不是靠相互之间的RPC或REST应用层交互，而是通过共享数据库做了一部分交互。这不应该被提倡！微服务的边界划分，应该遵循 `数据+数据的操作` 封装在一起的原则。数据在谁那里，操作就应该在谁那里。**如果两个应用需要共享DB，那就应该考虑应用上的拆分是否合理，是不是应该合并到一个应用里？！**

- 依赖关系：删除对 `memcloud-data-dao` 的依赖，增加对 `memcloud-dns` 的应用层交互。



- **对DB的依赖**： 目前mem-alert具体哪里依赖了 `memcloud-data-dao` 呢?



小技巧: 在 `memcloud-alert/pom.xml` 里删除对 `memcloud-data-dao` 的依赖，就能在Eclipse工具里快速展示哪些代码对它依赖了。

```
<groupId>io.memcloud</groupId>
<artifactId>memcloud-alert</artifactId>
<version>0.2.0</version>
<name>memcloud-alert</name>

<dependencies>

    <dependency>
      <groupId>io.memcloud</groupId>
      <artifactId>memcloud-data-mongodb</artifactId>
      <version>0.2.0</version>
    </dependency>

    <!--
    https://github.com/downgoon/memcloud/issues/6
    a better way is RPC/REST, not direct dao
     -->
    <!--
    <dependency>
      <groupId>io.memcloud</groupId>
      <artifactId>memcloud-data-dao</artifactId>
```

```
      <version>0.2.0</version>
      <exclusions>
        <exclusion>
          <artifactId>memcloud-data-model</artifactId>
          <groupId>io.memcloud</groupId>
        </exclusion>
      </exclusions>
    </dependency>
     -->
</dependencies>
```

4 附录1：依赖关系配置

4.1 mem-dns

```
<parent>
  <groupId>io.memcloud</groupId>
  <artifactId>memcloud</artifactId>
  <version>0.2.0</version>
</parent>


<artifactId>memcloud-dns</artifactId>
<name>memcloud-dns</name>
<version>0.2.0</version>
<packaging>war</packaging>


<dependencies>
    <dependency>
      <groupId>io.memcloud</groupId>
      <artifactId>memcloud-data-dao</artifactId>
      <version>0.2.0</version>
    </dependency>
</dependencies>
```

4.2 mem-alert

```
<parent>
  <groupId>io.memcloud</groupId>
  <artifactId>memcloud</artifactId>
  <version>0.2.0</version>
</parent>

<artifactId>memcloud-alert</artifactId>
<name>memcloud-alert</name>
<version>0.2.0</version>
<packaging>jar</packaging>


<dependencies>
    <dependency>
      <groupId>io.memcloud</groupId>
      <artifactId>memcloud-data-mongodb</artifactId>
      <version>0.2.0</version>
    </dependency>
</dependencies>
```

4.3 mem-boos

```
<parent>
  <groupId>io.memcloud</groupId>
  <artifactId>memcloud</artifactId>
  <version>0.2.0</version>
</parent>


<artifactId>memcloud-boos</artifactId>
<name>memcloud-boos</name>
<version>0.2.0</version>
<packaging>war</packaging>


<dependencies>
   <dependency>
      <groupId>io.memcloud</groupId>
      <name>memcloud-dns</name>
      <version>0.2.0</version>
```

```
    </dependency>

    <dependency>
      <groupId>io.memcloud</groupId>
      <name>memcloud-alert</name>
      <version>0.2.0</version>
    </dependency>

    <dependency>
      <groupId>io.memcloud</groupId>
      <artifactId>memcloud-cas</artifactId>
      <version>0.2.0</version>
    </dependency>
</dependencies>
```