两种架构的比较及最终选择

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | data-shared | | MVC | |
| 比较方面 | **pros** | **cons** | **pros** | **cons** |
| 安全性 |  | 个人信息数据和视频数据集中存放。数据中心易成为被攻击的对象，发生单点失效的可能性更大 | 控制器对数据的控制比较分散，受到攻击后影响较小 |  |
| 可用性 | 数据比较集中 心跳易于实现对整个数据中心的监测。  向外封装了主从数据库，易于完成错误恢复实现高可用性 |  |  | 数据分散，心跳对异常的监测较为困难 |
| 互操作性 | 没有明显区别 | | | |
| 可修改性 |  | 数据的消费者和生产者在数据方面紧密耦合 不容易修改 | 模块内聚性强、耦合度低，容易修改而不影响其他模块 |  |
| 性能：负载 | 封装了数据的获取、存储操作，易于实现支持并发数据存取，保证系统在高负载情况下正常运行 | 数据中心架构不当易成为负载的瓶颈 |  |  |
| 性能：容量 | 没有明显区别 | | | |
| 性能：实时性 | 没有明显差别 | | | |
| 可维护性 | 视频资源、数据较为集中，对数据、资源的维护更加简单 |  | 模块内聚性强、耦合度低，易进行维护 |  |
| 可测试性 | 数据生产者和数据消费者之间的逻辑借助数据中心完成隔离，数据生产者和数据消费者的相应模块可以分别测试 |  | 模块内聚性强、耦合度低，桩测试和驱动测试易于实施 |  |
| 易用性 | 没有明显差别 | | | |
| 开发难度 |  | 框架资料少，无可借鉴成熟案例，框架搭建出现失误的可能性大 | 有较多成熟高效的MVC框架，开发速度快、开发成本低 |  |

整体比较