链滴

服务器规约和应用分层

作者：xixiaoming
原文链接：https://ld246.com/article/1542682336382
来源网站：链滴

服务器规约

- 【推荐】调小TCP协议的time_wait等待时间.

说明: 基于TCP的HTTP协议由服务端关闭TCP连接,在连接以四次握手结束后,该连接仍会进入time_wai状态等待一段时间后才真正关闭.在高并发访问情形下,会因处于等待关闭的连接数太多而无法建立新连接,为了缓解或根治这一问题需要在HTTP服务器上调小此等待值.调小此等待值对于优化HTTP服务的请求处理能力和降低服务器负载效果明显.在Linux服务器上请通过变更/etc/sysctl.conf文件去修改缺省值.

- 【推荐】调大服务器所支持的最大文件句柄数(File Descriptor,简写为fd).

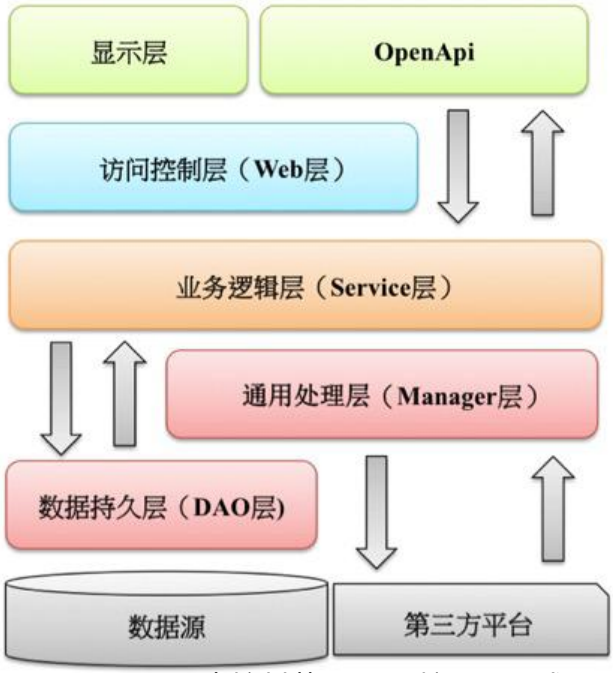
说明: 主流操作系统的设计是将TCP/UDP连接采用与文件一样的方式去管理,即一个连接对应于一个fd集团使用主流的Linux服务器默认所支持最大fd数量为1024,当并发连接数很大时很容易因为fd不足而现"open too many files"错误,导致新的连接无法建立. 建议将Linux服务器所支持的最大句柄数调高倍(与服务器的内存数量相关).

- 【推荐】给JVM设置-XX:+HeapDumpOnOutOfMemoryError参数,让JVM碰到OOM场景时输出dmp信息

说明: OOM的发生是有概率的,甚至有规律地相隔数月才出现一例,出现时的现场信息对查错非常有价值.

应用分层

- 【推荐】推荐如下分层结构,图中默认上层依赖于下层,箭头关系表示可直接依赖,如:OpenApi层可以赖于Web层,也可以直接依赖于Service层,依此类推:



- OpenApi层:可直接封装Service接口暴露成HSF接口,或者通过Web封装成http接口.
- 显示层:模板渲染层.当前主要是velocity渲染,JS渲染,JSP渲染等.
- Web层:主要是对访问控制进行转发,各类基本参数校验,或者不复用的业务简单处理等.
- Service层:相对具体的业务逻辑服务层.
- Manager层:通用业务处理层,它有如下特征:

1. 对第三方平台封装的层,预处理返回结果及转化异常信息;
2. 对Service层通用能力的下沉,如缓存方案、中间件通用处理;
3. 与DAO层交互,对DAO的业务通用能力的封装.

- DAO层:数据访问层,与底层Mysql、Oracle、Hbase、OB进行数据交互.
- 第三方平台:包括其它部门HSF开放接口,基础平台,其它公司的HTTP接口.
- 【参考】(分层异常处理规约)在DAO层,产生的异常类型有很多,无法用细粒度异常进行catch,使用catch(Exception e)方式,并throw new DaoException(e),不需要打印日志,因为日志在Manager/Service一定需要捕获并打到日志文件中去,如果同台服务器再打日志,浪费性能和存储.

在Service层往上抛的同时就必须使用日志,因为是RPC调用,可能本地出错,未必能正常反馈到业务端.这就必须尽可能带上参数信息,相当于保护案发现场,并打印异常堆栈.

如果Manager层与Service同机部署,日志方式与DAO层处理一致,如果是单独部署,则采用与Service一的处理方式.

Web层绝不应该继续往上抛异常,因为已经处于顶层,无继续处理异常的方式,如果意识到这个异常将导页面无法正常渲染,那么就应该直接跳转到友好错误页面,尽量加上友好的错误提示信息.

openApi层要将异常处理成错误码和错误信息方式返回.

- 【参考】分层领域模型规约:

1. DO(Data Object):与数据库表结构一一对应,通过DAO层向上传输数据源对象.
2. DTO(Data Transfer Object):数据传输对象,Service和Manager向外传输的对象.
3. BO(Business Object):业务对象.可以由Service层输出的封装业务逻辑的对象.
4. QUERY:数据查询对象,各层接收上层的查询请求.注:超过2个参数的查询封装,禁止使用Map类来传输.
5. VO(View Object):显示层对象,通常是Web向模板渲染引擎层传输的对象.