# DCOS环境应用改造规范

目录

[DCOS环境应用改造规范 1](#_Toc445394881)

[前言 3](#_Toc445394882)

[一、DCOS平台架构概要 4](#_Toc445394883)

[1、技术架构： 4](#_Toc445394884)

[2、功能架构： 6](#_Toc445394885)

[二、应用开发规范 7](#_Toc445394886)

[1、应用预编译 7](#_Toc445394887)

[2、热点数据缓存 10](#_Toc445394888)

[3、应用无状态化 11](#_Toc445394889)

[4、应用上配置的定时任务需与应用分离 12](#_Toc445394890)

[5、应用使用固定的服务监听端口 12](#_Toc445394891)

[6主机名和服务端口获取 13](#_Toc445394892)

[7、数据库连接池 14](#_Toc445394893)

[8、应用日志 14](#_Toc445394894)

[9、应用中间件选型 15](#_Toc445394895)

[10、应用环境配置规范 15](#_Toc445394896)

[11、硬件负载 16](#_Toc445394897)

[12、不要在容器中存储数据 16](#_Toc445394898)

[13、不要发布两份应用 17](#_Toc445394899)

[14、不要创建超大镜像 17](#_Toc445394900)

[15、不要在容器中运行多个进程 17](#_Toc445394901)

[16、不要在镜像中存储凭据、使用环境变量 18](#_Toc445394902)

[17、使用非root用户运行 18](#_Toc445394903)

[18、不要依赖IP地址 18](#_Toc445394904)

[19、服务间接口调用使用TCP和HTTP协议 19](#_Toc445394905)

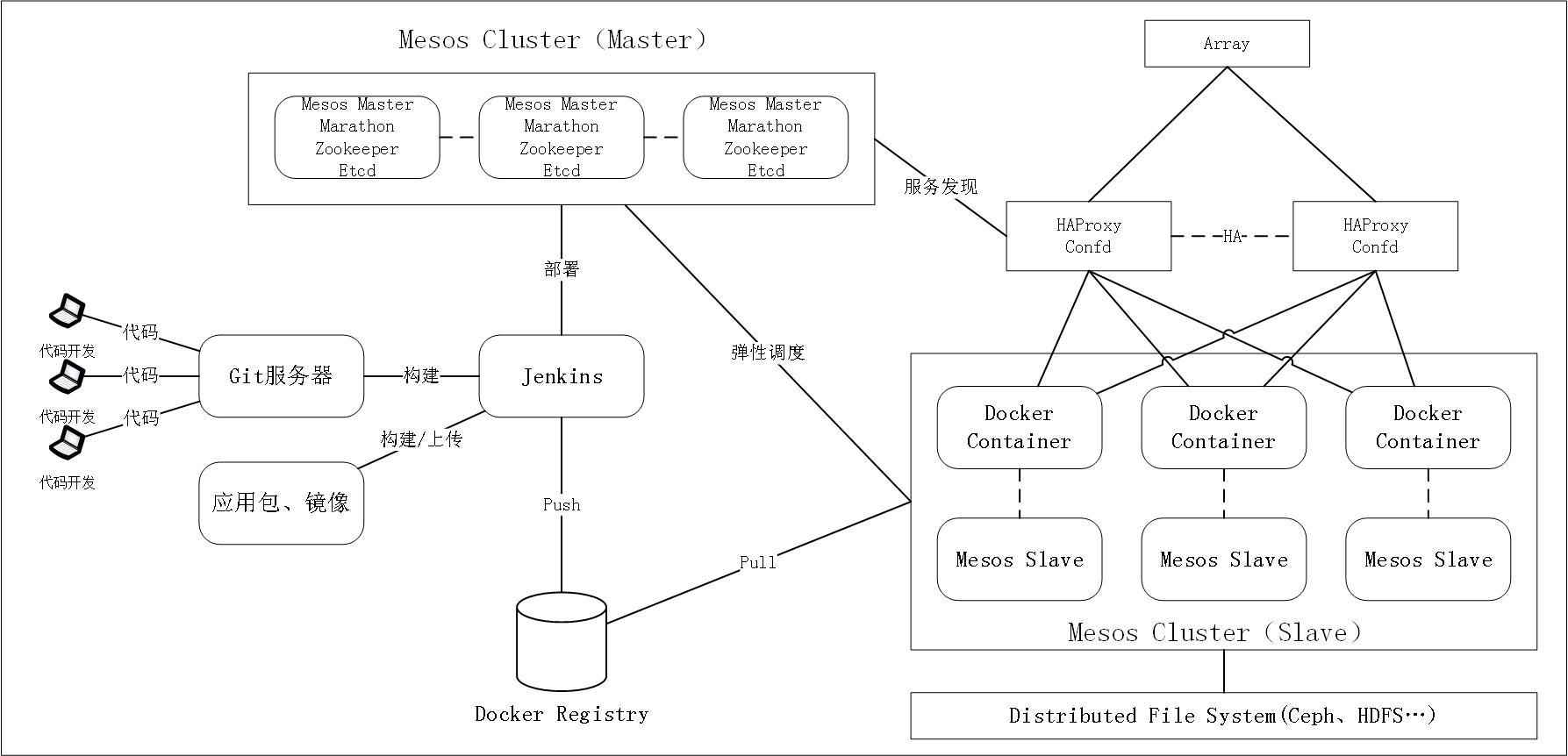
# 前言

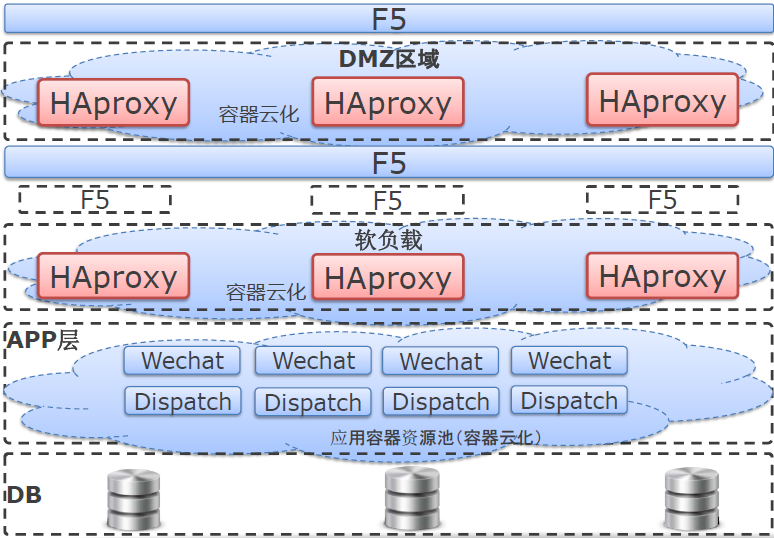
传统数据中心历经了传统的小机时代、X86化到IAAS资源池化的发展进程，但是整体IT架构依然处于“烟囱式”的架构模式。整个IT运营管理面临着应用部署手动化、应用扩容缩容非实时性、前端负载均衡技术无法实现应用动态引流、无法持续集成与部署，无法快速迭代应用程序、监控方法分散不统一、容灾环境各异的难题，同时传统的数据中心高可用能力比较差，这些都增添了IT运营的成本及工作量。在这种环境下，新型数据中心操作系统DCOS对改造滞后的IT系统架构提供了针对性很强的解决方案！

## 一、DCOS平台架构概要

### 1、技术架构：

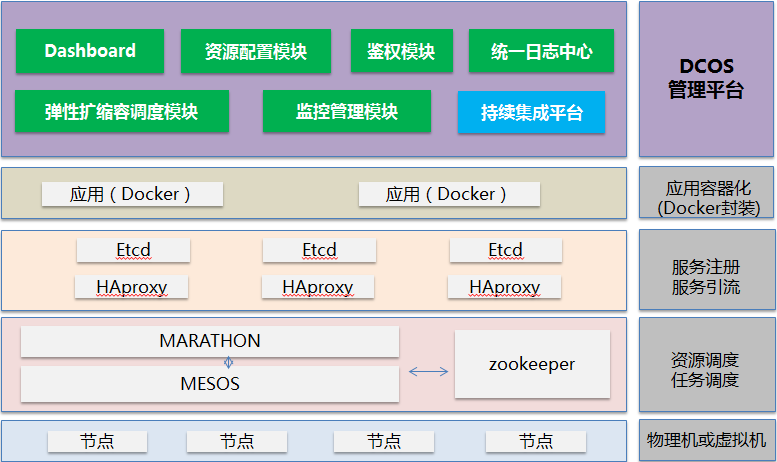
DCOS项目的整体架构采用了以容器为基础封装各类应用和运行环境，以Mesos、Marathon为核心实现容器资源的分布式调度与协调,以Haproxy、Etcd为辅助实现服务注册引流。架构如下：





### 2、功能架构：

以开源技术Mesos 、Marathon 、Docker、HAProxy为引擎，在其上开发了DCOS控制台、资源配置模块、鉴权模块、统一日志中心、弹性扩缩容调度模块、监控管理模块、持续集成平台。



## 二、应用开发规范

### 1、应用预编译

1）问题描述

根据J2EE规范，JSP主要是HTML文件，在它里面包含着Java代码用来和其他的系统组件进行交互以及动态的显示信息。规范规定所有的J2EE编译应用服务器应当支持JSP，客户请求一个特定的JSP，将：

* 转换JSP从HTML格式成为servlet类型的Java类（Java源格式），用简写的JSP符号代替完全符合规定的Java语法。
* 将新产生的Java源文件编译成.class字节码形式。
* 在新编译的类上执行适当的接口方法并且对客户端请求返回响应。

虽然从发展的观点来看对于在表示层内管理动态HTML的产生这是最好的途径，但它影响到服务器的运行时间环境，要求JSP被解析、转变成Java代码，并且在它去处理一个特定的客户端请求之前被编译。对最终用户明显的影响是，一个响应将会被延迟知道给定的JSP文件被编译通过。考虑到一个特定的用户请求可能用到两个或多个JSP文件，因此编译状态必需的时间增加了很多倍。

2）建议方案

一般预编译JSP的方法是使用Java命令行。它允许开发者在发展阶段和在部署前解决编译时间问题的时候编译需要的JSP文件。它也为生产系统提供一个有能力实现JSP预编译的管理员。这种用法的主要好处是：

* 文件可以被预编译一次然后可以被多次部署。（这不被服务器实例的重复利用所影响）
* 编译时的例外可以被预先解决而不影响部署。
* 类可以通过集群部署。

为了调用命令行JSP编译器，你必须确定下面的内容：

* PATH环境变量必须包含你机器上安装的J2SE1.3包的二进制目录（例如，/opt/j2se/1.3.1/sdk/bin或者c:sunsoftj2se1.3.1sdkin），以获得JVM运行时的支持。如果你打算使用javac作为你的JSP编译的Java编译器，要确定PATH包含全部Java 1.3的软件开发工具包（SDK）的二进制目录，并且不仅仅是JRE（Java Runtime Engine，Java运行时间引擎），因为没有编译器和JRE关联。如果你打算使用一个编译器而不是javac（例如 Jikes），也要为那个编译器确定在PATH中包含正确的目录。
* 设置Java系统类加载（classpath）路径应该来自$middleware\_home的lib目录，通过在产品库目录下默认建立（例如，/opt/bea/wlserver6.1/lib/weblogic.jar，当应用程序中有同名JAR文件时，需将JAR包拷贝到/opt/bea/wlserver6.1/lib/目录下，并以该目录下的JAR包为准）。此外，请确定在JSP编译阶段中你可能需要的参考类（JAR或类文件）也在你的类路径中。如下面的jsp示例：



### 2、热点数据缓存

1. 问题描述

热点数据指应用系统在运行过程中经常使用的数据，传统方式是应用系统在启动时通过数据库加载到应用实例缓存中，提高数据的访问速度，但往往因为加载的数据过多而导致系统启动过慢。

1. 建议方案

建议应用在设计时将热点数据存放到缓存服务器（如Redis、Memcached）中 ,在启动时避免执行加载缓存动作，提高应用系统启动速度，相关热点数据可通过手动或自动方式更新到缓存服务器中，应用通过访问缓存服务器获取相关热点数据。

### 3、应用无状态化

1. 问题描述

有状态应用是指用户在登陆后会有一定的会话信息（也就是应用中的session对象）存储在实例中，如果该实例异常停止，该用户的会话信息就会丢失。

1. 建议方案

建议应用开发设计时将用户的会话信息保存到缓存服务器（如Redis、Memcached）中，根据用户的唯一标识session\_id来对缓存服务器进行存取用户的会话信息，这样会话信息与应用实例分离，实现应用无状态话。

1. 实施样例

应用服务器本地会话保存改为使用key-value数据库（redis）存储。Session的读取依然由中间件Session Manager进行自定义管理。

例如：tomcat中加载tomcat-redis-sessionmanage-1.2.jar 、commons-pool-1.6.jar 、jedis-2.1.jar，

tomcat/conf/context.xml中加入如下配置：

<Manager className=“com.orangefunction.tomcat.redissessions.RedisSessionManager”

maxInactiveInterval=“60” //当前会话的失效时间

sentineMaster=“自定义” //与redis配置保持一致

### 4、应用上配置的定时任务需与应用分离

1. 问题描述

传统方式定时任务都与应用打包在一起，通过修改配置文件的特定配置项来控制定时任务是否启动，这样就会出现应用的差异化，在系统运行时就需要去手动配置。

1. 建议方案

建议将定时任务与应用分开，不要打包到一个应用包中。

### 5、应用使用固定的服务监听端口

1. 问题描述

对于标准的服务只需提供一个对外服务端口，而有些应用在运行过程中会自己产生一些服务端口对外服务，而且这些端口是随机产生，给管理带来不便。

1. 建议方案

建议在应用设计时尽量不要在应用运行时产生额外的服务端口，如果必须使用则需让端口固定下来并体现在系统集成文档中。

### 6主机名和服务端口获取

1. 问题描述

应用在运行时需要获取应用实例所在的主机名和实例服务端口信息，而DCOS采用容器化技术，获取到的主机名和实例服务端口信息为虚拟信息，导致一些应用程序运行异常。

1. 建议方案

建议尽量避免使用主机名和服务端口信息，如必须使用可通过环境变量 HOST和PORT获取相关信息。

### 7、数据库连接池

1. 数据库连接池最大值：中间件中数据库连接池的配置需数据库的最大process来进行计算。如数据库使用oracle，设置的连接数为1000，通过评估得出该数据库只为某个应用使用且该应用最大需要使用20个容器，则平均每个容器中的中间件数据库连接数为1000/20=50个。
2. 数据库连接池初始值:因为DCOS在进行弹性扩容时会有大量应用实例同时启动，而每个实例的数据库连接池在初始时都会创建一定链接，如果连接池的初始值设置过大会造成数据库无法及时响应大量并发连接而导致数据异常，为避免该情况，建议初始值设置小一些，建议设置为3个以下。

### 8、应用日志

1）应用日志文件命名规则：[应用名]\_[容器ID]\_catalina.out\_[date]

2）应用日志统一输出到/app/logs目录下

3）每个容器中应用日志文件个数最大为5个，每个日志文件容量最大为200M

4）日志在测试环境中开启调试开关，但在生产环境中必须关闭调试开关

### 9、应用中间件选型

1）动态应用建议使用tomcat轻量级中间件

2）静态应用建议使用nginx高性能WEB 服务器

3）缓存服务器建议使用Redis高性能缓存服务器

### 10、应用环境配置规范

1. 问题描述

应用系统在开发过程中需要分别在开发环境、测试环境、预生产环境和生产环境，而对于不同环境，应用只是某些配置文件的配置不同（如数据库连接地址，访问接口地址）不同，而程序代码相同，在不同环境中部署时就要进行重新打包。

1. 建议方案

建议应用针对不同环境准备相应的配置文件，并全部打包到应用包中，我们以web.xml为例，命名规则如下：

* 开发环境—dev\_web.xml
* 测试环境—test\_web.xml
* 预生产环境—pre\_web.xml
* 生产环境—pud\_web.xml

并将文件存放路径写入系统集成文档中。

### 11、硬件负载

1）要求采用轮询策略

2）健康检测需采用HTTP协议

3）TIMEOUT、FIN\_TIMEOUT时间不得小于HAProxy配置的TIMEOUT、FIN\_TIMEOUT时间

### 12、不要在容器中存储数据

1）由于容器是一次性的，当容器被停止、销毁或替换时，应用在容器中存储的数据同样会被销毁。

2）如果应用需要存储数据，需存储在共享数据存储中。

### 13、不要发布两份应用

1）每个容器中只能部署运行一个应用，不能部署多个应用到一个容器中。

### 14、清除不必要的包和文件

1）容器的一个显著特点是秒级启动，一旦制作了一个超大的镜像将难以分发。在镜像中应该确保仅有运行应用/进程的必需的文件和库存在，其他不必要的包、文件等建议应用打包前进行清除。

### 15、不要在容器中运行多个进程

1）容器能完美地运行单个进程（http守护进程，应用服务器，数据库），但是如果你不止有一个进程，管理、获取日志、独立更新都会遇到麻烦。

### 16、不要在镜像中存储凭据、使用环境变量

1）不要在应用中写死服务的IP和端口（如数据库IP和端口、服务接口IP和端口），采用域名的方式。

2）环境变量应该以传参的形式传入容器，而不是直接写死在容器中，需要的环境变量可写入系统集成文档中。

### 17、使用非root用户运行

1）docker容器默认以root运行。

2）随着docker的成熟，越来越多的安全默认选项变得可用。请求root是危险的，可能无法在所有环境中可用。所以镜像应该使用USER命令来指令容器以一个非root用户来运行。

### 18、不要依赖IP地址

1）每个容器都有自己的内部IP地址，如果你启动并停止它地址可能会变化。

2）如果应用或微服务需要与其他容器通讯，由于容器IP是不固定的，应用依赖IP地址将无法与其他容器通讯，可以选择更合适的方式与其他容器进行通讯。

### 19、服务间接口调用使用TCP和HTTP协议

1. 问题描述

服务间接口调用使用非标准协议进行交互(如EJB调用使用t3协议)，无法通过负载均衡器进行有效负载。

1. 建议方案

建议服务间接口调用通过TCP和HTTP协议进行交互，这样可使用负载均衡器进行有效负载。