第一篇

1、介绍

从计算机诞生开始，就伴随着计算机应用程序的演变。简短的回顾历史，我们可以清楚的看到应用程序发生的巨大变化。上世纪 70 年代中期，随着个人 PC 机的爆炸式增长以及程序员的崛起，让计算机的计算能力得到了大跨越的提升，个人 PC 机上可以运行非常复杂的应用程序。

进入上世纪 80 年代，随着 Bulletin Board System（简称：BBS）电子公告板系统的兴起，它可以为广大 PC 机用户提供基本的在线服务，如在线聊天、电子邮件、消息发送和文件下载。由于受到那个时代计算机网络传输速度的限制，在线服务的响应速度慢，交互体验差是最大的通病。

进入 90 年代中后期，随着万维网的出现，计算机的计算能开始进入快速提升阶段，加之网络基础设施的持续完善，计算机网络技术也随之发展起来，这让 Web 网站可以提供功能多元化和更为复杂的在线服务，直到今天，我们所看到的互联网（或云）开发的在线服务应用程序。

在这段计算机技术快速成长的时间里，计算机软件到底发生了哪些变化？从历史的发展中，我们可以看到，应用程序本身没有发生本质的变化（程序=数据结构+算法），变化的是软件的供需方式发生了

改变。现在，应用程序消费者不需要再在他们的 PC 机上下载和安装特定的应用程序，即可获得软件所提供的计算服务。在云计算技术的支持下，消费者（企业或个人）只需要使用 Web 工具（浏览器）访问并登录软件提供商的 Web 系统，通过简单的配置，就可以获得自己所需应用程序服务。这种通过网络即可使用软件的服务，即使 SaaS（软件即服务）。

在本篇文章中，我们将着重介绍 SaaS 架构设计，并围绕 WHAT（是什么？）、WHY（为什么？）、WHERE（在哪里？）和 HOW（怎么样？）这四个问题，对以下的几点进行阐述：

2、什么是 SaaS 平台?

在你决定实施 SaaS 品台架构设计前，你有必要先了解 SaaS 平台是什么。从宏观的角度来看，SaaS 是一种软件应用程序交付方式，软件提供商集中化托管一个或多个软件应用程序，并通过互联网向租户体用这些软件应用程序。从分类上看，SaaS（软件即服务）也是云计算重要的一部分。目前国内主流的云服务提供商如阿里云、百度云、腾讯云等，为广大用户提供了不同业务需求的云服务，它们大致可以分为以下几类：

1、基础设施即服务：如 CPU、Network、Disk 和 Memory 等

2、平台即服务：如阿里云服务器和云数据库等

3、软件即服务：阿里短信、阿里邮箱等

4、数据即服务：如阿里云对象存储，七牛云存储等

5、其他软件服务：机器学习、人工智能等

SaaS 应用程序的任何更新或者修复漏洞操作都是由软件提供商负责实施和处理的，由于租户是通过互联网获取软件服务，所以租户端无需下载任何的升级包或者修复补丁，是一种开箱即获取最新软件产品的服务方式。

通过对什么是 SaaS 的介绍，接下来，我们了解一下选择 SaaS 作为软件架构来设计产品的一些理由。

3、为什么选择 SaaS?

我们将从不同的角度来阐述几个为什么选择 SaaS 的理由。透过对这些因素的分析，为你是否需要将自己的软件 SaaS 化提供一定的参考依据。

3.1、消费者角度

获取软件服务的方式足够简单，SaaS 也许是迄今为止使用软件最简单的方式之一，租户只需要动动鼠标和键盘，即可在几小时甚至几

分钟内获得一个大型的软件服务。相比于传统使用软件的方式，租户省去了研发、部署、运维等一系列繁复的过程，且获得软件的时间和费用成本都大幅度降低。

3.2、商业角度

SaaS 可以使用跨地域、跨平台的软件服务。与此同时，软件服务商可以统一对软件进行版本管理，这将带来以下几点好处（包括但不限于）：

1、缩短产品上线时间：多端适配，统一版本，统一更新

2、降低维护成本：不需要同时维护多个版本的软件实例，运维压力减小

3、容易升级：由于版本得到有效控制，一次升级，即可覆盖所有租户端

4、SaaS 的特性和优势

我们将 SaaS 应用程序与传统的桌面应用程序做一个水平的对比，部署一个 SaaS 产品将可以获得以下的几点优势。

4.1、简单

SaaS 化的产品通过互联网向租户提供软件服务，随着 Web 技术（如 jQuery、Node.js）的进步，Web 页面的交互体验度大幅度提

升，交互更流畅、更人性化。与传统的桌面应用程序的人机交互效果相差无几。

4.2、经济实惠

SaaS 化产品可以为租户提供弹性的付费方案，如按日、按月、按年、按使用人数或者按使用量进行计费，它将给租户提供更经济的使用软件的财务预算表。

4.3、安全

使用 SaaS 产品无需担心数据安全问题，这好比将钱存入银行一样安全。相较于企业内部部署的软件系统而言，SaaS 产品具备更高的安全保障能力，因为软件提供商具有更多软件安全防护的技术资源、人力资源和财政资源。

4.4、兼容性

与传统软件相比、SaaS 软件的兼容性更好，它没有传统软件的多本版维护问题和操作系统兼容问题。在 SaaS 软件中，租户用户在使用软件的过程中，几乎上感觉不到软件发生了改变。当租户用户登录到系统上时，就已经获得了最新版本的软件。

5、SaaS 软件的适用范围

SaaS 产品具有广泛的适应范围，特别是与其他云产品（如 IaaS(基础设施即服务)和 PaaS(平台即服务)）配合使用时这种能力表现尤为突出，例如阿里云之类的云计算技术允许你配置可托管的 Web 站点、数据库服务器等。你只需要打开浏览器并登录到阿里云控制台，通过操作对应的控制面板，即可获得相关的软件服务。

从理论上讲，SaaS 可以将任何的软件 SaaS，下面列举一些通用的分类供大家参考：

1、Office 在线办公类 SaaS 产品

2、电子邮件和即时消息类 SaaS 产品

3、社交媒体类 SaaS 产品

4、第三方 API 类 SaaS 产品

5、安全和访问控制类 SaaS 产品

6、机器学习类 SaaS 产品

7、人工智能类 SaaS 产品

8、地理位置服务类 SaaS 产品

9、数据流和数据检索类 SaaS 产品

6、SaaS 产品的天生缺陷

6.1、软件控制权

与企业内部部署的软件不同，由于 SaaS 软件被击中托管在服务提供商的 Web 服务器中，所以租户无法控制所有的软件应用程序，SaaS 化的软件比企业自行部署的软件获得的控制权更少，租户可操作的自定义控制权极度有限。

6.2、消费者基数小

由于 SaaS 软件是将一套应用程序共享给一个或者多个租户共同使用，这种共享的消费方式还未被大多数的消费者所接受。同时，受制于市场环境的影响，目前还有大多数的软件还未 SaaS 化。

6.3、性能瓶颈

共享应用程序必然会带来服务器性能的下降、如计算速度、网络资源、I/O 读写等都将面临严峻的考验。在性能方面，企业内部部署的“独享模式”的应用程序比 SaaS 软件的“共享模式”略胜一筹。

6.4、安全问题

当租户在选择一款 SaaS 产品时，产品的安全性将会被放置在第一位进行考虑。如数据的隔离、敏感数据的加密、数据访问权限控

制、个人隐私等问题。在 2018 年 5 月 25 日，GDPR(General Data Protection Regulation)《通用数据保护条例》出现之后，越来越多的人开始重视数据安全问题。如何最大程度的打消租户的这一顾虑，需要服务提供商加强对自身信誉度的提升，以赢得租户的信赖。

7、SaaS 产品的核心组件

不同类型的 SaaS 产品，由于要面对不同的用户愿景，可能在功能和业务上会有所不同，但任何一个 SaaS 产品，都具备以下几个共同的核心组件。

7.1、安全组件

在 SaaS 产品中，系统安全永远是第一位需要考虑的事情，如何保障租户数据的安全，是你首要的事情。这如同银行首选需要保障储户资金安全一样。安全组件就是统一的对 SaaS 产品进行安全防护，保障系统数据安全。

7.2、数据隔离组件

安全组件解决了用户数据安全可靠的问题，但数据往往还需要解决隐私问题，各企业之间的数据必须相互不可见，即相互隔离。在 SaaS 产品中，如何识别、区分、隔离个租户的数据时你在实施 SaaS 平台架构设计时需要考虑的第二个问题。

7.3、可配置组件

尽管 SaaS 产品在设计之初就考虑了大多数通用的功能，让租户开箱即用，但任然有为数不少的租户需要定制服务自身业务需求的配置项，如 UI 布局、主题、标识（Logo）等信息。正因为无法抽象出一个完全通用的应用程序，所以在 SaaS 产品中，你需要提供一个可用于自定义配置的组件。

7.4、可扩展组件

随着 SaaS 产品业务和租户数量的增长，原有的服务器配置将无法继续满足新的需求，系统性能将会与业务量和用户量成反比。此时，SaaS 产品应该具备水平扩展的能力。如通过网络负载均衡其和容器技术，在多个服务器上部署多个软件运行示例并提供相同的软件服务，以此实现水平扩展 SaaS 产品的整体服务性能。为了实现可扩展能力，就需要 SaaS 展示层的代码与业务逻辑部分的代码进行分离，两者独立部署。例如使用 VUE+微服务构建前后端分离且可水平进行扩展的分布式 SaaS 应用产品。对于可扩展，还有另外一种方式，即垂直扩展，其做法比较简单，也比较粗暴：通过增加单台服务器的配置，如购买性能更好的 CUP、存储更大的内存条、增大带宽等措施，让服务器能够处理更多的用户请求。但此做法对于提升产品性能没有质的改变，且成本很高。

7.5、0 停机时间升级产品

以往的软件在升级或者修复 Bug 是，都需要将运行的程序脱机一段时间，等待升级或修复工作完成后，再重新启动应用程序。而 SaaS 产品则需要全天候保障服务的可用性。这就需要你考虑如何实现在不重启原有应用程序的情况下，完成应用程序的升级修复工作。

7.6、多租户组件

要将原有产品 SaaS 化，就必须提供多租户组件，多租户组件是衡量一个应用程序是否具备 SaaS 服务能力的重要指标之一。SaaS 产品需要同时容纳多个租户的数据，同时还需要保证各租户之间的数据不会相互干扰，保证租户中的用户能够按期望索引到正确的数据，多租户组件是你必须要解决的一个问题。其余的组件都将围绕此组件展开各自的业务。

第二篇

1、Saas 概述

1.1、软件发展的四个阶段

（1）项目式软件开发阶段 -- 做项目

-- 依客户需求定制开发

-- 存在重复开发，开发成本过高的问题

（2）套装式软件开发阶段 -- 做产品

-- 将软件作为产品开发，满足相似需求的客户

-- 不可能通过产品满足所有用户的需求

（3）平台化软件开发阶段 -- 做平台

-- 业务驱动基于基础平台的软件开发

-- 软件的升级和运营维护成本越来越高

（4）社会化软件大开发阶段 -- 做服务

-- 以服务为导向的软件开发运营模式

-- Saas 模式应运而生

1.2、软件即服务

Saas 是一种软件交付模式,将软件以服务的形式交付给用户，用户不再购买软件，而是租用基于 Web 的软件，并按照对软件的使用情况来付费

Saas 由应用服务提供模式发展而来。

Saas 与 Asp 的相同点：通过互联网提供，运营商负责软件的管理和维护。

Saas 与 Asp 的不同点：ASP 仅对用户提供定制化的，一对一的服务方式，Saas 一般以一对多的方式提供服务，Saas 支持可配置性和可伸缩性。

1.3、Sass 与云计算

云计算 -- 基于互联网的新计算模式

主要可分为三个层次：

Iaas -- 基础设施即服务

Paas -- 平台即服务

Saas -- 软件即服务

Saas 可以基于 Pass 构建，也可以直接构建在 Iaas 上

Saas 的发展催生了对 Paas、Iaas 的需求

Saas 为云计算提供了一种应用模式

Saas 将云计算的能力推向了最终用户

1.4、Saas 的特性

互联网特性 -- Saas 应用一般通过互联网交付，用户仅需要浏览器或联网终端设备就可以访问应用

多租户特性 -- 通过多租户模式实现多种使用方式，以满足不同用户的个性化需求

按需服务特性 -- 支持可配置性和按使用付费，按用户需求提供服务

规模效应特性 -- 一般面向大量用户提供服务，以取得规模效应和效益

1.5、Saas 成熟度模型

定制开发的 Saas 应用 -- 每个租户一套（不同实例）

可配置的多租户 Saas 应用 -- 每个租户一套（相同实例）

单实例支持多租户的 Saas 应用架构 -- 单个实例

支持可伸缩性的多租户 Saas 应用架构 -- 集群部署

1.6、Saas 的优势

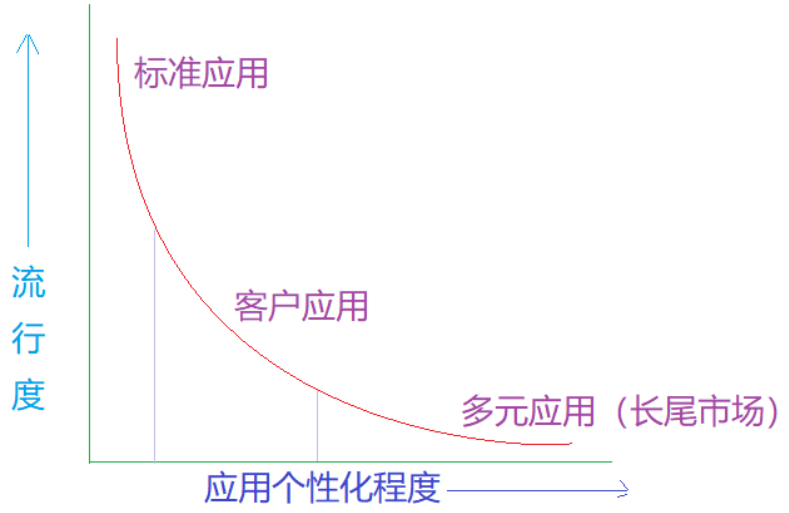
服务的理念符合软件的本质

按需服务，契合用户的需求

软件统一管理，易于升级和维护

随处可用，支持移动办公

有利于抓住长尾市场



用户使用 Saas 降低了使用软件的成本，增强了业务变化的灵活性

企业采用 Saas 扩大了用户范围，提高资源的使用效率，提升收益

2、实现 Saas 模式的挑战

2.1、多租户模式

多租户模式是指 Saas 应用可以同时为多个用户提供有差别服务的软件使用模式，每个用户（租户）都可以不受其他用户影响的访问应用，并可以定制应用的某些属性

多租户模式是 Saas 的核心优势

对多租户模式的理解是 Saas 成败的关键

多租户模式对 Saas 应用的功能性和性能有更高的要求

多租户模式需要全新的软件设计开发方法

2.2、用户需求获取

按需服务是 Saas 应用的核心理念 -- 多租户 Saas 应用应尽可能的满足不同用户的个性化需求，通过多个租户向用户提供有差别的服务

用户需求的获取 -- 研究用户需求的建模、分类和实现方式，是设计开发多租户 Saas 应用的基础

用户需求的描述 -- 如何对用户需求模型进行有效的整合和挖掘，从而合理设置 Saas 应用的业务灵活性，是 Saas 模式的一大挑战

2.3、多租户个性化与可配置性

多租户的个性化需求 -- Sass 应用只有通过多租户模式尽可能满足不同类型用户的个性化需求，才能在商业模式上取得成功

个性化需求配置工具 -- Saas 应用需要提供配置方法及工具，以便租户根据个性化需求定制 Saas 应用的功能及非功能属性，可配置性一般体现在数据、功能、界面、流程、安全、性能等多个方面，配置工具应该功能强大，同时又易于使用

2.4、高效率运行和可伸缩性

高效率运行 -- Sass 模式将软件服务通过互联网交付给用户使用，在短时间内大量用户并发访问的情况下，Saas 应用能否高效运行对服务可用性和用户体验有重要意义

可伸缩性 -- 随着业务的增长，在服务的用户量和访问持续上升的情况下，如何实现 Saas 应用的可伸缩性，保持可接受的性能和可用性，以保证用户的良好体验，对 Saas 应用能否取得成功至关重要

2.5、数据独立和事务性

数据以托管方式存储和管理 -- 在 Saas 模式下，全部用户的数据存储和管理都由服务运营商负责，数据是用户关注的重点

数据独立性 -- 数据空间独立性（物理隔离）、数据结构独立性（逻辑隔离）

事务处理 -- 对于有状态 Saas 应用，在多租户用户并发访问的情

况下，实现有效发的数据事务处理机制，保持业务与数据的一致性是一项重要的挑战

2.6、资源共享与隔离性

资源共享 -- 多租户模式下，物理资源共享可以自然实现，应重点实现租户之间基础设施、数据资源的共享，共享可以提高资源使用效率，共享提升了 Saas 应用的可用性和性能

资源隔离 -- 在实现资源共享的同时，针对用户对性能、安全性和其他方面的需要，实现资源的隔离性与独立性，是 Saas 模式有待解决的问题

2.7、安全性保障

安全性是用户对软件的首要需求，Saas 模式为软件的安全性提出了新的要求，由于 Saas 通过网络交付给终端用户，在处理远程应用访问和远程数据传输方面需要更高的安全性，由于数据层的基础设施可能同时提供给来自跨组织的多租户使用，数据的安全性必须得到完全的满足。

数据安全性包括 CIA -- 机密性、完整性、可用性

2.8、服务质量保障

用户的非功能需求 -- 非功能需求的满足是实现良好用户体验的基础，而用户体验对 Saas 应用的成功至关重要

服务质量要求 -- 用户的非功能需求主要体现在对服务质量的要求上，例如服务的可用性、可靠性、响应时间、处理速度等

服务等级协议 SLA -- 规定了服务质量，如何在系统资源有限的情况下，通过对资源、程序代码、配置等多个方面的优化，为用户提供有 SLA 保障的服务，是 Saas 模式的发展方向

2.9、租户/用户管理与计费

租户/用户管理 -- 租户合理设置（功能、权限、数据模型等方面）、动态管理维护租户信息、动态管理用户并为用户分配资源

计费策略 -- 以按使用付费为原则，应实现计时、计次、计数据量、计功能点等多种策略，应实现计费策略的灵活组合和改变

3、Saas 架构参考

参照 ITA 的描述方法，Saas 的主技术架构通过一系列的视图从不同角度来描述：

概念视图 -- 关键元素和元素之间关系的高层次的总览

逻辑视图 -- 表示了主要功能组件和它们在系统中的关系

实现视图 - 表示特定的实现组件和它们之间的关系

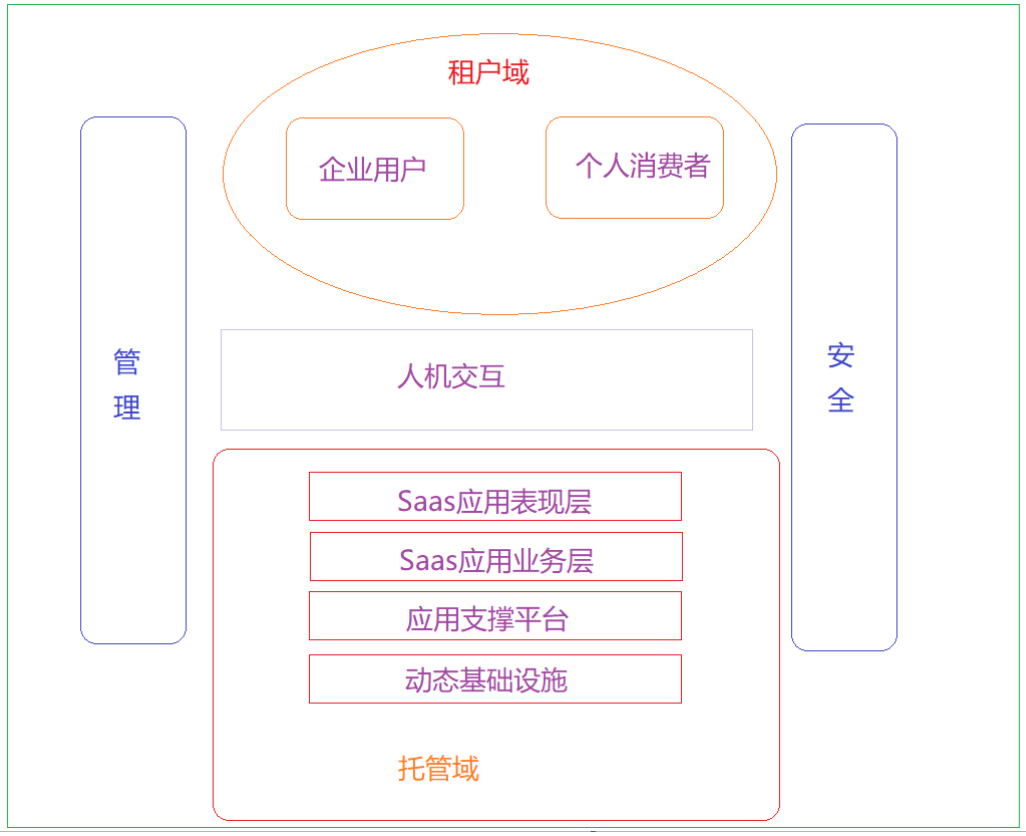
3.1、概念视图

租户域 -- 包括应用的租户和最终用户

托管域 -- 应用的提供和运营平台

管理 -- 负责应用的管理事宜

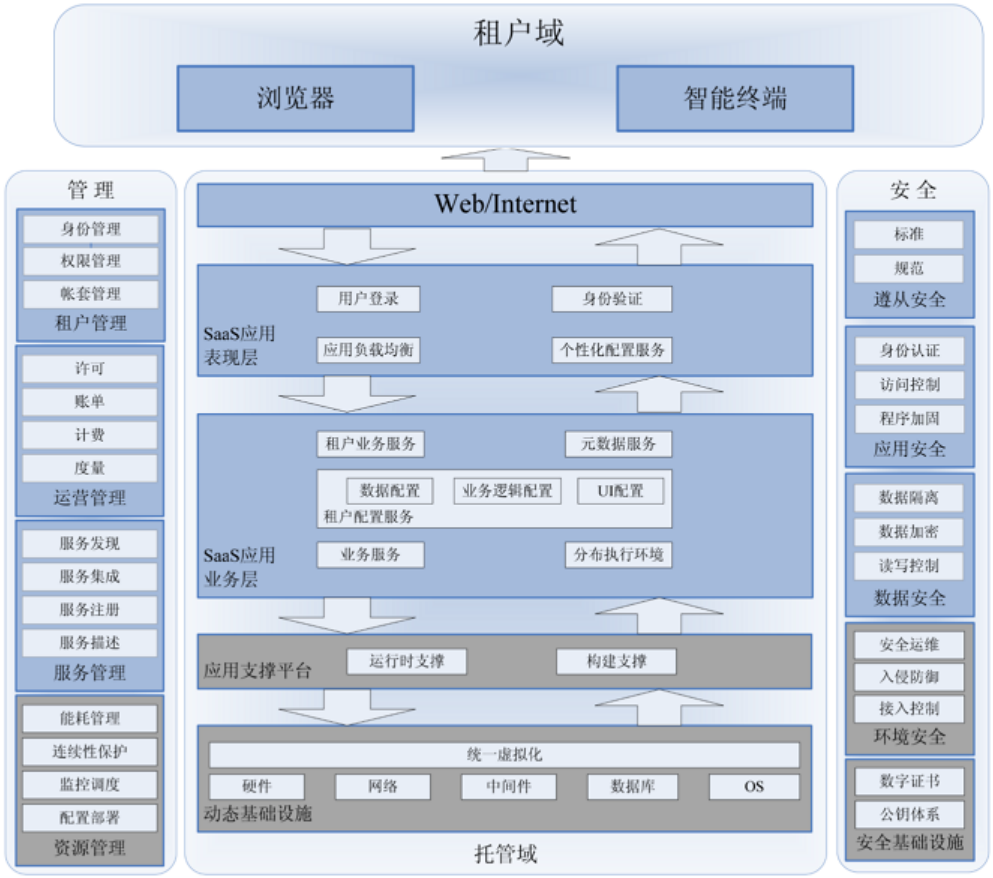
安全 -- 负责应用的安全事宜



3.2、逻辑视图

租户域 -- 由浏览器+智能终端组成，功能是信息输入输出，简单计算、存储能力，普适感知能力

Saas 应用表现层 -- 负责应用的页面表现，支持单点登录和统一身份认证，支持应用层负载均衡和执行请求自动转发，支持租户个性化配置



Saas 应用业务层 -- 通过业务服务和租户服务实现个性化服务，通过元数据服务和租户配置服务实现租户业务和数据等方面的定制，通过分布执行提供高可用性和高性能服务

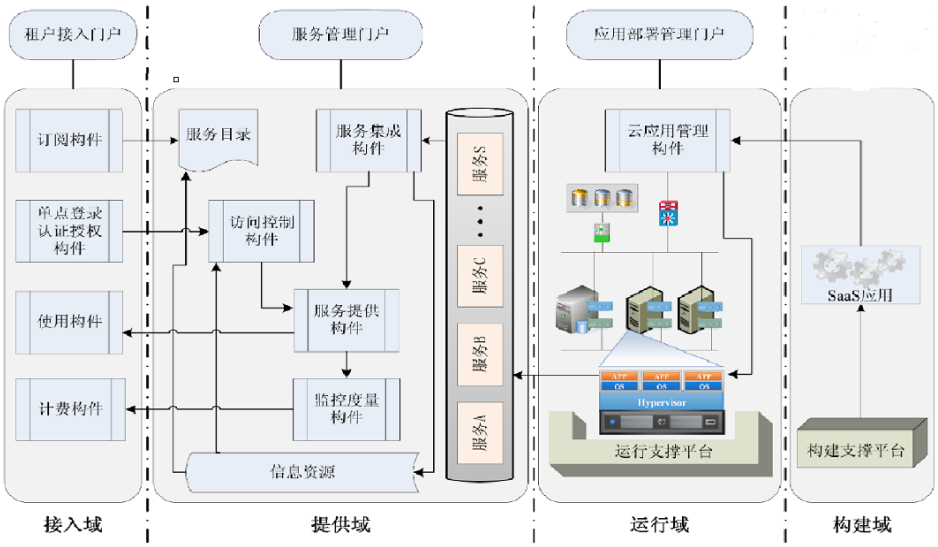
应用支撑平台 -- 运行支撑（提供信息集成、数据划分优化模式、存储框架，并行计算框架等方面的支持）、构建支撑（提供开发环境、存储模型、计算模型等方面的支持）

动态基础设施 -- 提供硬件、网络、中间件、数据库与 OS 的支持，提供支持统一虚拟化的计算、存储、网络通信与交换能力

租户管理、运营管理、资源管理、服务管理

应用安全、数据安全、环境安全、基础设施

3.3、实现视图



4、构建 Saas 应用的关键技术

4.1、A 级：定制开发的 Saas 应用

为租户单独定制开发应用，与传统软件相比，主要体现在软件租用付费商业模式和托管运营模式的区别，一般不涉及大的技术架构变化，为提高应用运行效率，需要更有效地整合硬件资源。

4.2、B 级：可配置的多租户 Saas 应用

采用统一开发的模式，所有租户使用相同的程序代码，但各租户分别部署程序实例，与 A 级模型相比，降低了定制开发的软件研发成本，关键在于通过元数据实现应用的可配置性

元数据 -- 用于描述数据的数据，用户描述租户的个性化需求

4.3、C 级：单例支持多租户 Saas 应用

全部租户运行部署在单一程序实例上的同一套程序代码，最大限度的提高了系统资源的利用效率，同时降低程序代码升级维护的工作成本，单实例架构为应用开发带来了更大的复杂度，需要更多的初期投入

4.4、D 级：可伸缩的多租户 Saas 应用

租户通过负载均衡层访问以镜像方式部署在集群上的同一套程序代码实例上，通过负载均衡方法为应用提供可伸缩性，使应用在大量用户访问下保持可接受的应用可用性和性能，关键在于实现应用的可伸缩性，主要涉及负载均衡、资源管理等技术

4.5、如何选择适合的成熟度等级

A、B 级成熟度模型适合对 Saas 应用的隔离性安全性有较高的要求，并愿意为此付出较高价格的用户

C、D 级成熟度模型适合对价格较为敏感的中小型企业及个人用户

如果不同类型用户业务差异过大，则只适合 A 级成熟度模型

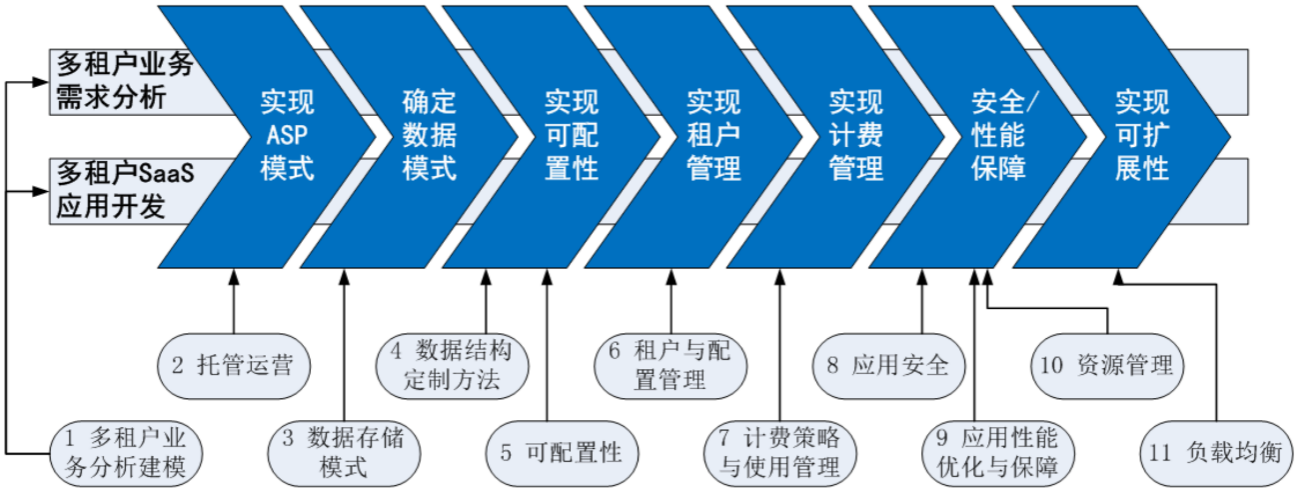
高级成熟度模型需要进行应用架构转变，会导致大量涉及开发工作，带来较大的初期投入，回报周期长，远期效益更好

选择适合的成熟度模型和数据存储方案，不同级别的成熟度模型和不同的数据存储方案各有侧重，适用于不同的场景和情况，根据用户需求、业务特性和自身条件合理选择

制定开发计划，确定关键技术，进行重点研发攻关，全面分析用户需求，整体设计，全面测试，设计/开发/测试/维护/客户支持等方面的思路转变对 Saas 的成功至关重要

4.6、基于软件产品构建 Saas 应用过程

面向 D 级（最高级）成熟度的 Saas 应用构建过程：



4.6.1、多租户业务需求分析与建模

全面考虑各类用户的不同需求，汇总全部业务需求情况，识别用户的共性需求与特定需求，对业务需求的通用性，必要性和实现代价分析与评估，按用户需求的相似度进行分析聚合，业务聚合模型是设计建立租户的基础，业务需求模型整合，业务需求模型是可配置性设计的基础

4.6.2、实现 ASP 托管模式

将软件转为基于网络访问的架构，为传统客户端软件定义访问交互接口，客户通过浏览器和联网终端设备即可访问使用，实现 ASP 托管模式软件本地运行，用户通过网络访问应用，整合硬件资源，性能优化，同一套硬件即基础设施之上，可以同时运行多个项目，支持多个租户使用

4.6.3、多租户数据存储模式

一般有三种模式：

- 完全独立模式（独立数据库实例模式）
- 部分独立模式（共享数据库实例，独立表集合）
- 完全共享模式（共享数据库实例，共享表集合）

按数据独立性、隔离性、安全性排序：

- 完全独立>部分独立>完全共享

按硬件共享程度、方案性价比排序

- 完全共享>部分独立>完全独立

完全独立模式 -- 租户拥有专属数据库实例，数据库实例之间逻辑独立，但可以部署在相同硬件上，性能比较容易控制，数据安全性，隔离性强，硬件开销较大，性价比差。

部分独立模式 -- 全部租户使用同一数据库实例，每个租户都有专属表集合，数据的安全性、隔离性与性能可控性弱于完全独立模式，硬件开销性价比有所提升，适合对数据独立性安全性有一定要求，但

预算有限的客户

完全共享模式 -- 全部租户共同使用同一数据库实例下的相同表集合，数据安全性、隔离性需要采用特定方法实现，性能可控性相对较差，最大化利用系统资源，性价比最强，适合对价格比较敏感的中小客户

4.7、数据结构定制方法

对完全独立与部分独立数据存储模式，通过每个租户定义不同表结构实现

对完全共享存储模式，通过特定方法实现预定义字段方法（aka、保留字段\固定扩展字段）、行转列方法（aka、数据字典/扩展字表/名称值对）、XML 字段方法、JSON 字段

4.7.1、预定义字段/保留字段

通过预定义扩展字段实现数据结构定制，非常易于实现，数据检索性能比较高，数据冗余大，扩展灵活性差

租户A1

患者信息表

ID	姓名	性别	保留字段1	保留字段2	保留字段3
1	张三	男	沈阳	210101198807062311	23
2	李四	女	北京	110105197309233242	38

元数据表

ID	表名	列名	列内容
2	患者信息表	保留字段1	患者城市
3	患者信息表	保留字段2	身份证号
4	患者信息表	保留字段3	患者年龄

租户B2

患者信息表

ID	姓名	性别	保留字段1	保留字段2	保留字段3
3	王五	男	门诊	普外科1	
4	赵六	男	住院	脑外科3	

元数据表

ID	表名	列名	列内容
5	患者信息表	保留字段1	患者类型
6	患者信息表	保留字段2	患者科室

4.7.2、行转列/名称值对

通过定义子表建立名称值对的方式实现数据结构定制，扩展灵活性强，但由于检索数据经常需要联合查询，性能较差

患者信息表

ID	姓名	性别
1	张三	男

患者信息扩展表

ID	患者ID	扩展字段	扩展内容
1	1	保留字段1	沈阳
2	1	保留字段2	210101198807062311
3	1	保留字段3	23

租户A1

元数据表

ID	表名	列名	列数据信息	列数据类型
2	患者信息表	保留字段1	患者城市	string
3	患者信息表	保留字段2	身份证号	string
4	患者信息表	保留字段3	患者年龄	int

患者信息表

ID	姓名	性别
3	王五	男
4	赵六	男

患者信息扩展表

ID	患者ID	扩展字段	扩展内容
7	3	保留字段1	门诊
8	3	保留字段2	普外科1

租户B2

元数据表

ID	表名	列名	列数据信息	列数据类型
5	患者信息表	保留字段1	患者类型	string
6	患者信息表	保留字段2	患者科室	string

4.7.3、XML 扩展字段（或 Json 字符串）

通过定义 XML（或 JSON）字段实现数据结构定制，扩展灵活性强，可以自由定制，对特定数据的检索实现较为复杂，额外开销较大

租户A1

患者信息表

ID	姓名	性别	XML数据
1	张三	男	<XMLData> <患者城市>沈阳<患者城市/> <身份证号>210101198807062311<身份证号/> <患者年龄>23<患者年龄/> <XMLData/>

租户B2

患者信息表

ID	姓名	性别	XML数据
3	王五	男	<XMLData> <患者类型>门诊<患者类型/> <患者科室>普外科1<患者科室/> <XMLData/>

4.8、租户可配置性

数据可配置性 -- 实现不同租户的定制化数据结构

功能可配置性 -- 通过租户管理订阅服务实现租户对原子功能的取舍

UI 可配置性 -- 租户可自定义界面风格及 LOGO

业务流程可配置性 -- 租户可根据业务需要自定义业务流程

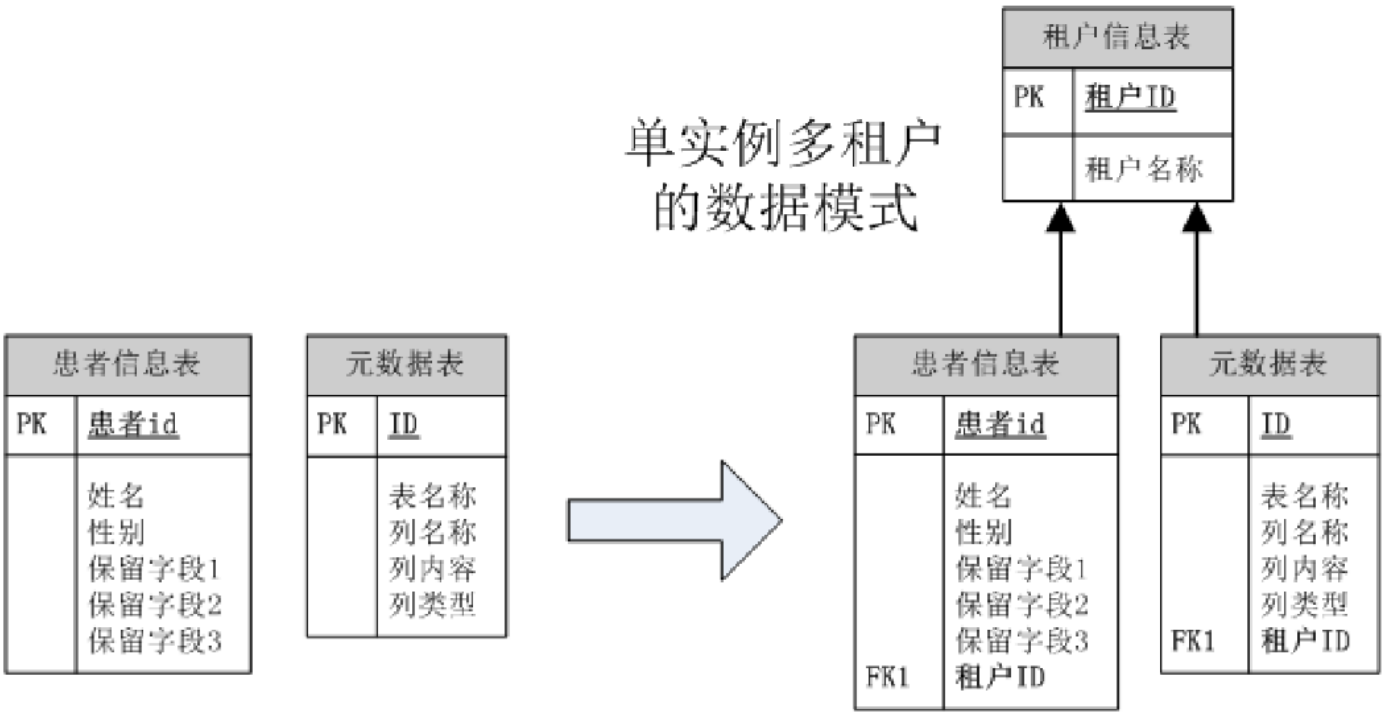
业务规则可配置性 -- 租户可在一定约束下设定灵活的业务规则

4.8.1、数据可配置性

完全独立或部分独立模式可直接实现数据可配置性

对完全共享模式，采用一种数据结构定制方法，同时为全部数据库表添加“租户 Id”字段

每个租户的数据检索添加“where 租户 Id = ?”的查询条件



4.8.2、功能可配置方法

将业务拆分成为多个功能包，通过提供功能包组合满足租户的不同需求，当预定义功能组合无法满足需求时，租户可定制功能包满足自身需求

示例：

医院管理信息系统 HIS

拆分功能包：门诊收费、门诊医令、住院管理、医嘱管理、电子病历、药物管理

提供功能包组合版本：门诊医保收费版、门诊划价收费版、门诊住院收费版、门诊住院全功能版

	门诊医保收费版	门诊划价收费版	门诊住院收费版	门诊住院全功能版
门诊挂号	●	●	●	●
医保收费	●	●	●	●
诊间医令		●	●	●
门诊收费		●	●	●
药房管理		●	●	●
出入院管理			●	●
住院收费			●	●
医生工作站				●
护士工作站				●
LIS				●
PACS				●

4.8.3、UI 可配置方法

租户为界面添加租户的LOGO，租户可选择界面主题，租户可定制界面控制的名称和位置等，可采用 Portal 等技术实现界面定制

4.8.4、业务流程可配置方法

根据业务不同，租户可能会有不同的业务流程需求，可采用工作流程引擎或业务流程引擎实现工作流的定制，工作流的定制应满足一定的业务约束，用户定制工作流之后，应经过一定的验证机制才能生效

4.8.5、业务规则可配置方法

业务规则可配置性将为业务带来极大的灵活性，可通过业务条件组合实现

4.8.6、 租户与配置管理

建立和维护租户信息（由服务提供者完成）

租户元数据配置、租户资源使用策略设置、租户访问权限设置、租户计费策略设置、租户管理者设定

管理各租户的用户信息（由租户管理者完成）

建立维护用户数据、用户权限与约束设置

4.8.7、计费策略与使用管理

计费策略

以按需提供、按使用情况付费为原则

以租户为单位，按功能和服务质量的不同制定不同的收费标准

提供按使用时间、使用次数、按数据量和按原子功能点计费等多种计费方式

使用管理

通过日志记录租户/用户的应用使用情况

对租户/用户的使用进行多维度度量（用户数、使用时间、次数、流量、功能点、组合维度）

按一定时段自动生成并发送账单

4.8.8、应用安全

应用层安全

通过权限控制确保用户操作的合法性，用户操作必须得到监控并且记录日志，需要标准化协议和立法的支持

数据层安全

合理设计数据存储方案保证数据隔离性，通过实时备份等机制保证数据可靠性，通过数据加密等手段保护数据的私密性，数据的非法访问必须被完全拦截

可信交互

在多应用集成并采用单点登录技术的环境下，必须遵循安全协

议实现应用之间的可信交互

4.8.9、性能优化与保障

数据层性能优化（合理建立索引、优化数据连接查询）

应用层性能优化（后台计算处理完成大量数据的统计等任务、异步通讯机制基于消息队列实现业务逻辑交互）

WEB 层性能优化（页面优化、提前加载内容、缩小 Cookie 体积）

通过性能监控与动态资源调配实现性能保障

4.8.10、资源管理

面向用户需求的系统资源优化分配（在满足用户需求的同时优化 Saas 应用的收益）、资源独立性与隔离性（租户之间、用户之间的资源使用情况互不影响）、动态资源调整（随时监控资源使用情况、根据实时情况动态调整资源分配方案、对资源过度使用进行限制）、资源管理实现 SLA 的基础和关键（完全支持 SLA 是 Saas 前进的方向）

4.8.11、负载均衡

应用层（基于租户特征进行访问请求优化分配、提高应用性能）

请求层（Session 复制、Session Sticky、共享 Cache）

网络层（虚拟连接交换、动态分配）

数据层（数据切分、读写分离）

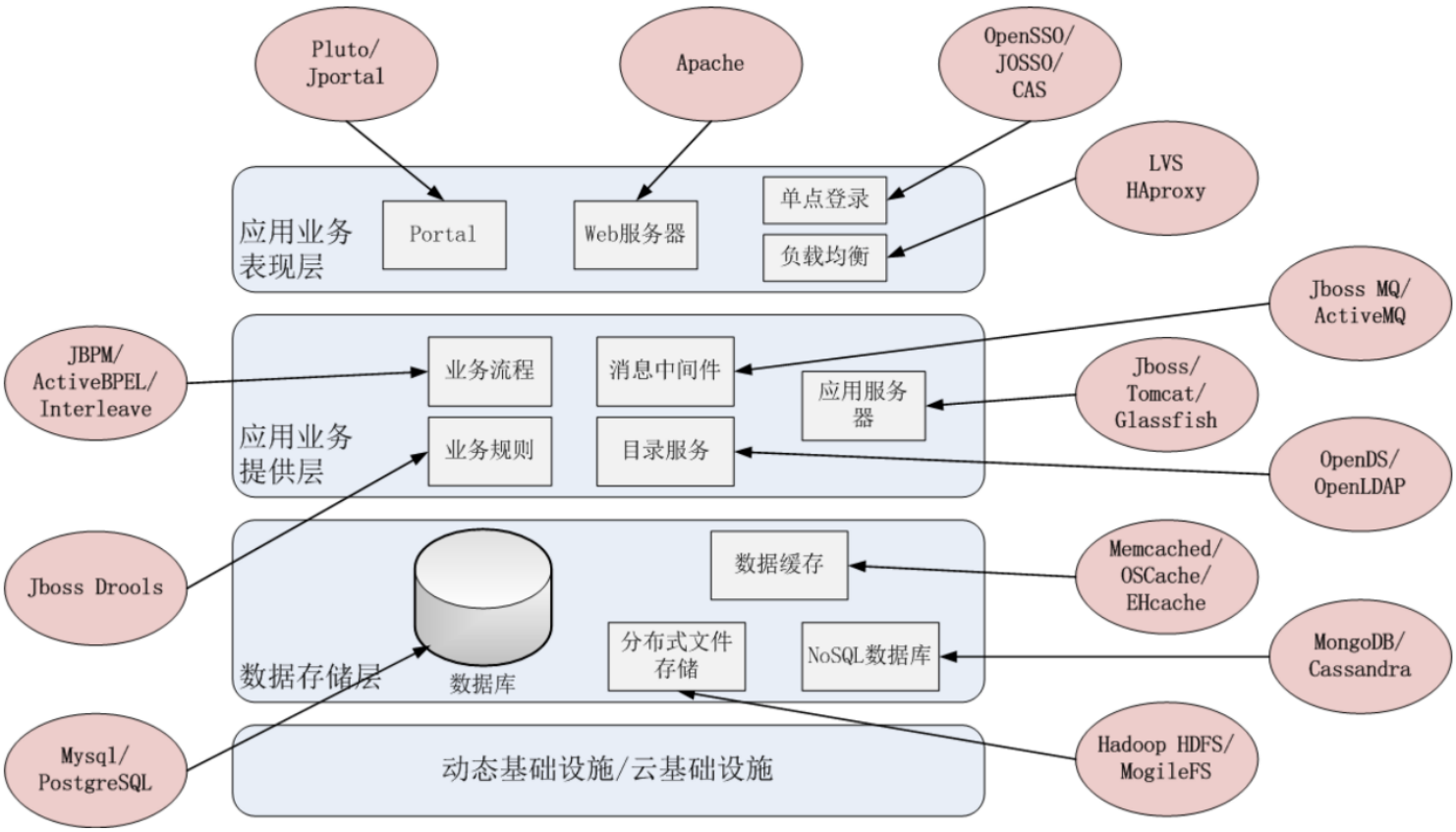
数据切分（将数据按一定规则进行划分，分别存储到不同终端，

然后按划分规则转发请求，通过水平扩展提高应用性能）

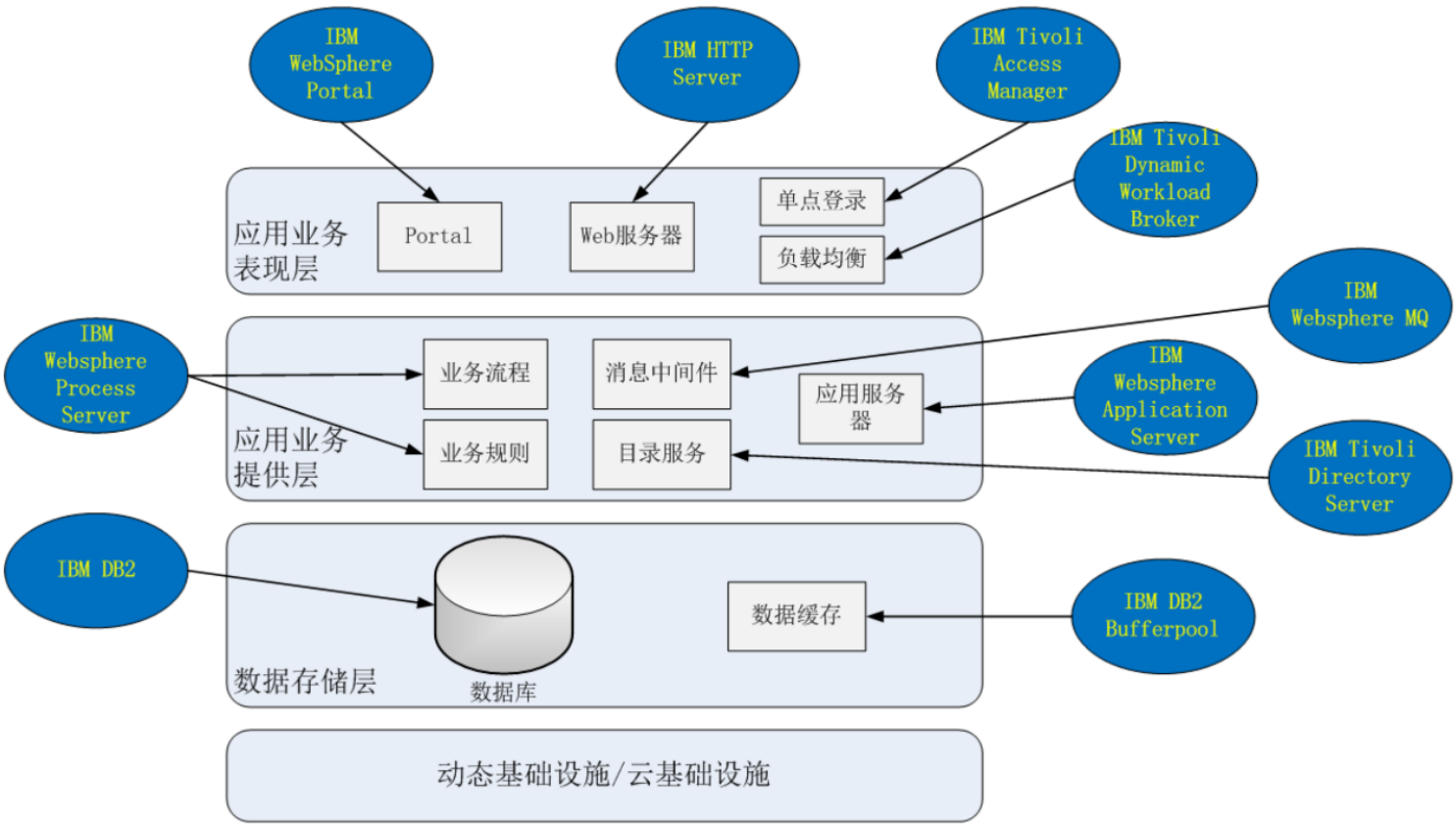
读写分离（将数据读写操作请求分别分配到不同服务器进行操作，服务器之间数据动态更新，适用于数据查询操作远大于存储操作的情况可提高应用性能）

5、Saas 应用参考实现方案

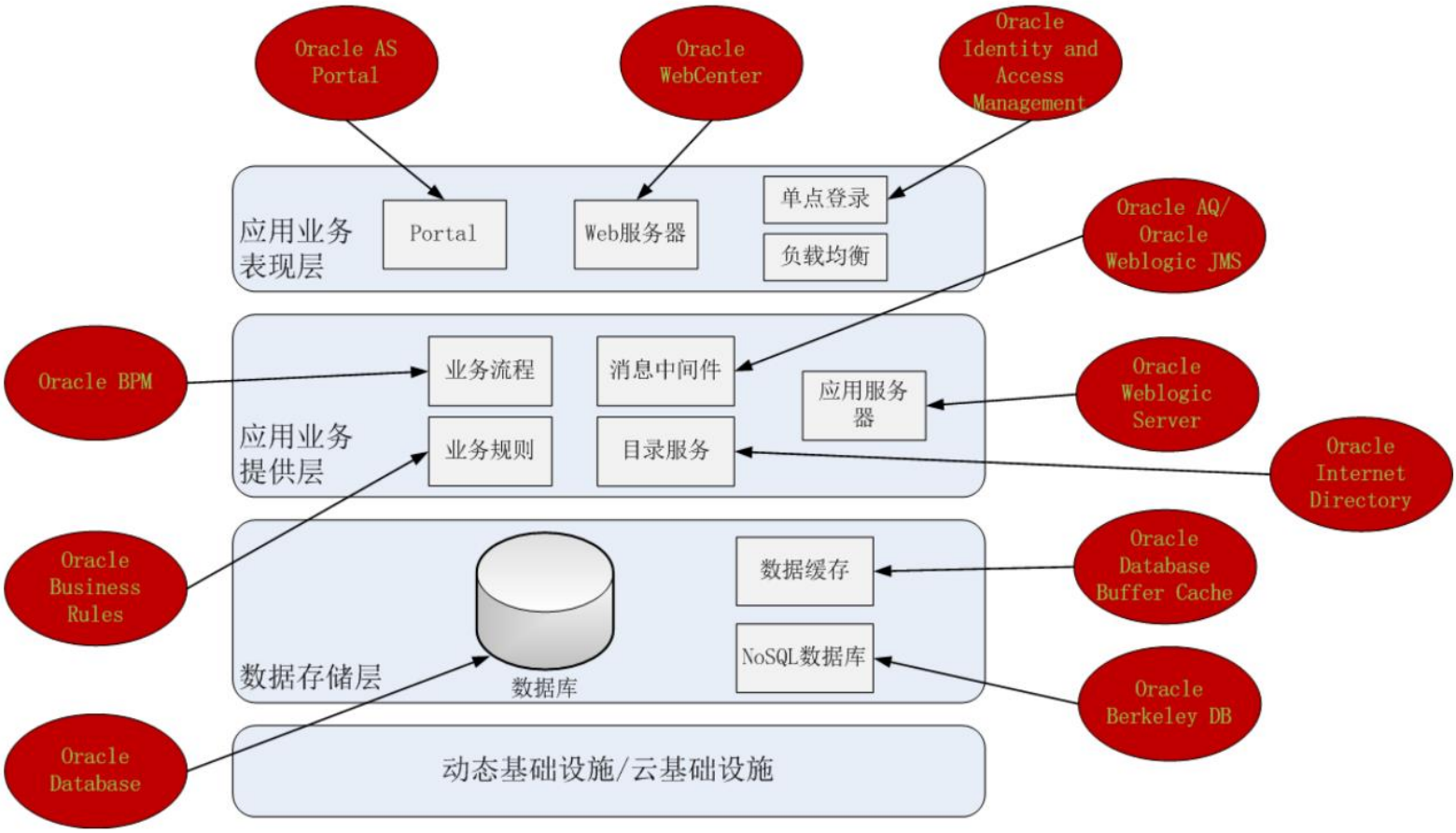
5.1、开源软件实现方案



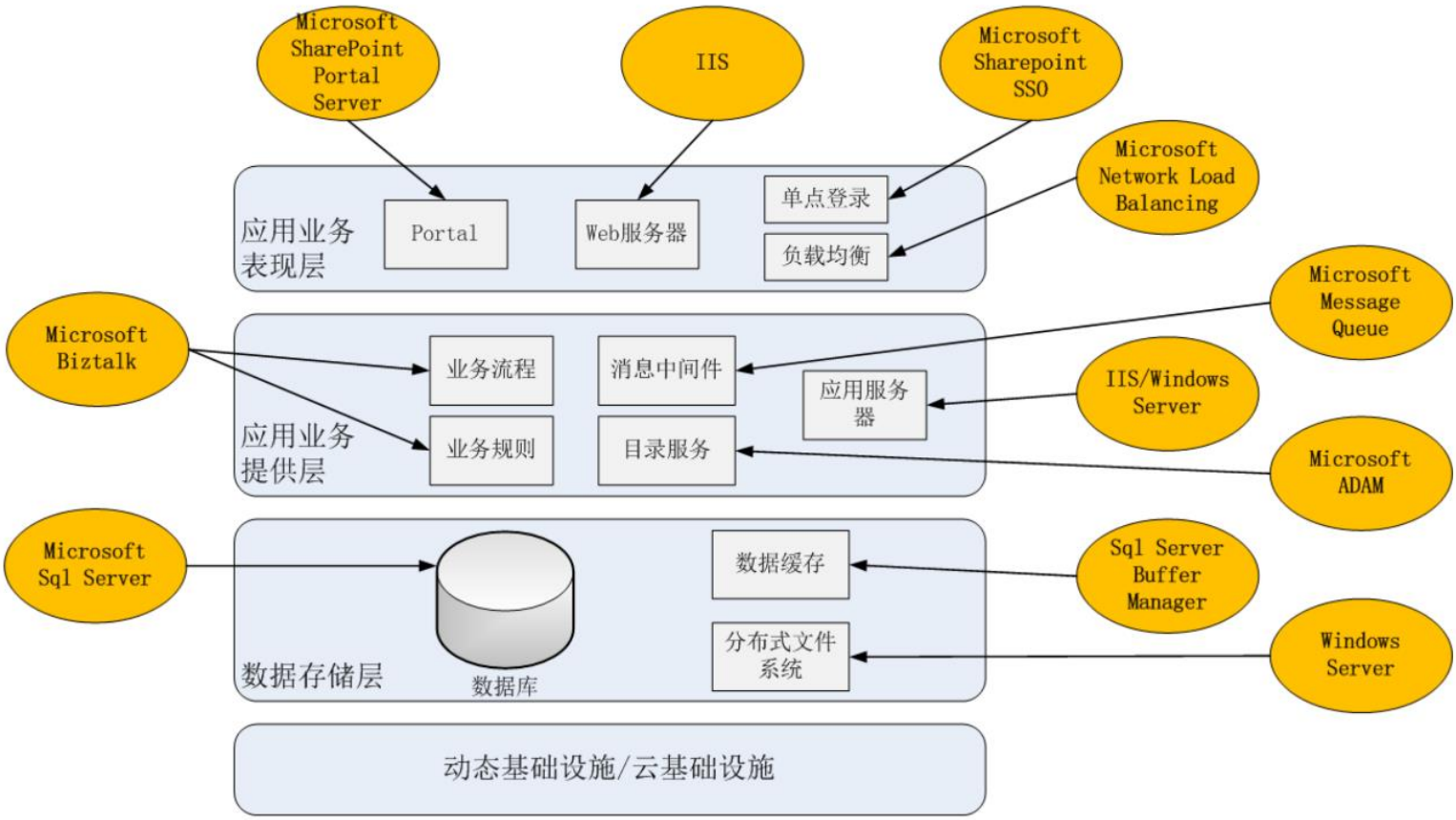
5.2、IBM 产品实现方案



5.3、Oracle 产品实现方案



5.4、微软产品实现方案


Microsoft SharePoint Portal Server
IIS
Microsoft Sharepoint SSO
Microsoft Network Load Balancing
应用业务表现层
Portal
Web服务器
单点登录
负载均衡
Microsoft Message Queue
Microsoft Biztalk
业务流程
消息中间件
应用服务器
IIS/Windows Server
应用业务提供层
业务规则
目录服务
Microsoft ADAM
Microsoft Sql Server
数据缓存
Sql Server Buffer Manager
分布式文件系统
Windows Server
数据存储层
数据库
动态基础设施/云基础设施

第三篇

实现 SaaS（软件及服务）架构所面临的三大技术挑战

在当下的世界中，软件应用无处不在，随处都可见电子产品的身影，一个没有任何电子产品的生活是什么样子让人很难想象。而支撑起这些电子产品的是无数个应用程序和其背后所隐藏的实现技术。

无论你需要做什么，都可以找到大量的应用程序供你选择。软件的本质工作是简化复杂的业务流程，让生活更简单，更有趣。然而，在享受软件带来便捷的同时，也伴随着一定的副作用，比如你需要安装大量的应用软件，且需要为这些应用软件支付一大笔费用，同时还需要关心软件升级维护的问题。为了最大程度的降低使用软件所带来的副作用，SaaS（软件及服务）应势而生。

在开始探讨本文的主题之前，先花一分钟了解什么是 SaaS 应用程序以及使用 SaaS 软件的优势。

1、什么是 SaaS 应用软件?

SaaS 是 Software as a Service（软件即服务）的缩写，它是云计算的主要体现形式之一，其他的还有诸如平台即服务（PaaS），基础设施即服务（IaaS）以及数据即服务（DaaS）等。简而言之，SaaS 是一种软件交付模式，旨在以“即用即付”服务的方式为客户提供软件服务，客户不需要安装软件便可获得应用程序提供的所有功能。

SaaS 是一种基于云计算的软件交付系统，客户只需要通过浏览器便可轻松订阅应用程序所提供的功能。这种“按需使用”或“软件+服务”的软件交付系统，被广泛应用于各大中小企业，组织中，以帮助他们降低软件的使用成本。当下，SaaS 软件正被用于各种行业的业务领域中，例如：客户关系管理软件（CRM），企业资源规划软

件（ERP），信息管理软件（MIS），人力资源管理软件（HRM），地理信息管理软件（GIS）以及学习管理软件（SIS）等等。

2、使用 SaaS 应用软件的优势

SaaS 软件交付模式的概念最早于 20 世纪 60 年代被提出，到 20 世纪 90 年代，随着互联网技术的不断发展，SaaS 的发展速度也随之加快，许多软件提供商开始通过互联网托管和提供软件服务，而 SaaS（软件即服务）这个术语也是在这一时期确定下来的。使用 SaaS 软件在以下几个方面可以获得较大优势：

在软件使用早期，可以大幅度降低硬件，带宽，安装和运营成本

可以以更小的人员和资金的开销，准确的预测业务数据，更快的完成软件的升级和维护工作

与传统软件相比，用户通过互联网获的软件服务，从而降低了管理软件所带来的安全风险

快速部署，快速接入。由于不需要自行安装和部署软件，只需要向服务提供商注册信息，并为订阅的服务付费，便可获得软件服务。从而减少了获得软件的等待时间

跨平台操作。借助互联网，客户可以在任何时间，任何地点，通过浏览器便可接入服务

任何有价值的东西，在其出现之前都会历经各种困难与挑战，没有什么事物是一蹴而就的。接下来，将介绍构建一个 SaaS 软件在系统测试，数据安全和升级维护三个方面面临的巨大挑战

3、实现 Saas 软件挑战一#测试

任何有价值的软件在交付之前，都需要进行大量的测试。这和我们买衣服一样，在确定付款前，都需要在试衣间试穿各种样式的衣服。SaaS 软件的测试与传统的软件测试基本相同，通过执行各种逻辑验证，确保作为服务提供的软件在质量上是否符合客户的需求。SaaS 软件的测试通常会包含数据安全测试，业务逻辑测试，数据集成测试，接口兼容测试，可伸缩测试和高并发测试。但 SaaS 软件的测试也有有别于传统软件测试的地方。

与传统软件开发相比，SaaS 软件的测试在测试周期和实施等方面都面临着诸多的挑战。尽管实现 SaaS 软件的技术框架会有所不同，但在测试环节都将面临一些常见的技术挑战。

3.1、安全性测试

SaaS 软件可以为使用者带来巨大的好处，但仍然有很多用户在质疑 SaaS 应用软件的数据安全。基于云计算应用的数据安全一直以来都是一个居高不下的热点话题。因此，针对 SaaS 软件的安全性测试必须慎重对待，需要有专门的测试策略和工具。

与其他任何的云计算平台一样，在云计算环境下维护数据的安全性和完整性具有很大的风险与挑战，和传统的软件相比，SaaS 应用软件的安全性测试更为复杂。在 SaaS 应用软件测试中，需要模拟多个租户下，不同安全级别的隐私要求，权限分配粒度，资源隔离等级和用户行为模式。传统的测试手段很难测试并发现 SaaS 软件中存在的安全漏洞，软件测试场景也很难完全发现漏洞并消除这些安全威胁。

3.2、可伸缩和高可用测试

可伸缩（也称可扩展）性是 SaaS 软件服务提供商重要的商业模型指标之一，要求 SaaS 应用软件可以根据客户量的大小进行水平方向的伸缩。简单来说，SaaS 平台可以根据当前用户量的多少，动态地增加或者减少运行实例的数量；而高可用（性能）是客户衡量 SaaS 软件好坏的一个重要指标。

如何成功的测试 SaaS 软件的可伸缩和高可用性，需要有专门的测试策略，才能组织出可用的测试场景，并且需要比传统软件测试更多的测试样本数据和测试标准。这些数据和标准需要仔细的考虑 SaaS 软件的应用场景，才可能被量化和设计。另外，还需要考虑如何在不同租户类型，不同的用户数量组合，不同使用环境（移动端，PC 端）的复杂条件下，对系统的性能，峰值和负载能力进行测试。

3.3、集成和开放 API 接口测试

SaaS 应用软件在一定程度上需要集成第三方的业务系统，同时还有可能需要开放一定的 API 接口，以支持从其他平台集成或迁移数据。在何种情况下，保护数据的安全性和完整性将给测试带来巨大的压力。在 SaaS 应用软件的集成测试和 API 测试中，需要对入站和出站数据进行验证，以及对所有 API 的功能，安全性，性能以及文档的完整性进行测试。即便是这样，你也很难提前组织起所有的测试场景，而且这个过程非常的耗时。也许第一版的 API 还没有测试完，新版本的 API 已经添加到测试列表。实现 SaaS 软件是一个不断迭代的过程，因此很难在短时间内一次测试就涵盖所有的 API。

4、实现 SaaS 软件挑战二#数据安全

对于 SaaS 软件而言，更为复杂和艰难的工作是如何保障 SaaS 平台中用户数据的安全。不管是大型企业还是小公司，他们对于数据的安全性要求都是一样严苛的。例如常见的 CRM 系统，HR 系统，ERP 系统和财务管理系统等，它们都存储了大量高度敏感的用户信息，如果 SaaS 软件的数据安全无法得到保障，不仅是客户的数据会遭受破坏，服务提供商的信誉也会收到严重的影响，甚至会伴随着相关法律的处理。因此，相比于传统的软件，SaaS 软件的安全性要求更高，系统结构更为复杂，实现难度也更大。

构建一个 SaaS 平台，在满足高性能和可伸缩的条件下，还需要着

力保障用户数据的高度安全，这主要体现在以下三个方面：

多租户数据隔离：多租户架构是 SaaS 软件的一个重要评定标准，如何对各个租户的数据进行识别，分割和存储需要在效费比，安全性和性能上取得一个平衡。

数据备份与恢复：由于各租户订阅服务的组合不同，使用软件的时间段也不尽相同，对不同租户的数据进行备份和恢复的难度也相当大。另外，在操作其中一个租户数据时，需要保障其他所有租户的数据不会受到影响。在发生系统故障时，还需要及时恢复租户数据，面对庞大的数据量，数据恢复的复杂度和难度都比单体应用高很多。

数据入站和出站校验：SaaS 软件允许用户在任何地点通过浏览器获取服务，这就要求 SaaS 软件在数据传输，用户输入，系统输出等环节有着更高的安全性要求。相比于单体架构的应用程序，需要更高级别的安全传输加密/解密手段，更细粒度的用户认证和鉴权措施。另外，对用户的日志的收集，追踪和审计工作比单体软件更难。

保护存储在 SaaS 平台中的数据安全，需要仔细的分析平台中每一个业务流程，细化权限下放的粒度，严格把控访问接入设置以及数据的存储规则。即便如此，由于 SaaS 系统通常需要面对比单体架构软件更为庞大的数据量，且业务流程更为复杂，更新周期相对较短，因此如何确保存储在 SaaS 平台中的数据不易被破坏或者泄露是一个永久存在的工作。

5、实现 SaaS 软件挑战三#升级维护

SaaS 应用软件免去了客户使用软件时的安装，维护，升级等工作，但这些繁杂的工作并未因为 SaaS 而消亡，而是转移到了 SaaS 软件服务商的手中，且工作变得更为复杂和艰难。

SaaS 软件需要全天候为客户提供可用的服务，因为你完全不知道客户会在什么时候登入系统开展工作。这就要求 SaaS 服务提供商在升级和维护软件的工程中不能影响当前用户使用软件。简单来说，SaaS 软件的升级维护不能采取“冷启动”的方式来完成，需要采用“热部署”的方式，让客户基本上感觉不到升级工作正在进行。反观传统软件上线或更新过程，往往会出现各种问题：不一致的运行环境，过多的人为干预系统的构建和部署，代码改动引起不可控的质量，向下不兼容，服务中断，更长的更新时间，数据丢失等问题。

对于 SaaS 系统，系统的升级维护工作不能暂停当前客户正在执行的业务，避免业务数据丢失，因此需要一种全新的软件发布机制，通过可视化，自动化的操作，实现持续，无缝，零重启的软件交付过程。在升级维护时，SaaS 软件主要面临以下几个挑战：

版本可回退：如果新上线的功能模块遇到重大问题，可以回退到之前的版本而不影响用户的正常业务。

系统向下兼容：新版本的系统需要尽可能的向下兼容旧系统的数据。在最坏的情况下，当升级过程发生时，用户正在使用旧版本提交数据，如果适配旧版本提交的数据，需要慎重考虑。

灰度发布：灰度发布包含两个方面：前后端灰度发布和移动端的灰度发布。

零重启：零重启要求在不终止服务的情况下完成系统的升级工作，这就要求 SaaS 平台具备热部署的特性，确保 SaaS 平台能够保持 7x24 小时的持续服务能力。

SaaS 软件易于使用是相对于软件使用者而言的，对于 SaaS 软件的提供者来说，软件的复杂性，安全性和可用性都面临者全新的技术难题，克服这些问题并不容易。作为 SaaS 软件的实现者，不能看着油漆是干燥的，就想当然的坐上去，在油漆真正干燥前，一切都是脆弱的，需要耗费大量的时间去处理诸多问题，才能让用户放心的坐上去。

市场对于 SaaS 软件的需求逐年递增，各种类型的 SaaS 软件产品也层出不穷，但真正成功的 SaaS 软件却凤毛麟角。究其原因，主要还是 SaaS 化的软件并不是简单的将传统软件改造成多租户架构那么简单，用户对于易用性，集成性，安全性，灵活性和定制性的要求越来越高，作为 SaaS 软件的实现者和提供者，需要付出比对待传统

软件更多的精力，更严苛的要求，更谨慎的考虑，才能正确认识到实现 SaaS 软件将要面临的技术难题与挑战。

基于云计算的 SaaS 软件交互模式，其多租户，多平台环境，高并发等特点给 SaaS 软件的实现带来了诸多的技术难点，作为 SaaS 软件的缔造者和提供者，需要全面和谨慎的考虑和处理这些技术问题，方可缩短软件承诺与用户期望之间的差距。提前识别并跟踪不断变化的需求与技术走向，谨慎的考虑架构与实现之间存在的问题，才能在打磨出一款有商业价值的 SaaS 产品。

外传篇

文档：SaaS 模式下多租户系统架构及关键技术研究

文档：SaaS 架构设计培训

文档：基于 AWS 的 SaaS 平台架构

文档：基于微服务架构 SAAS 产品研发实践

文档：平台级 SAAS 架构的基础--统一身份管理系统

文档：基于云计算架构的 SaaS 架构设计

文档：SAAS 后台权限设计案例分析

文档：关于 SaaS 平台中应对多租户模式的设计—权限设计

文档：大型企业基于 SaaS 的核心业务平台建设

文档：智能会话机器人：SaaS 平台的设计与思考