哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

软件工程

第三章　软件架构设计

3-3 SaaS与云端软件部署

徐汉川
xhc@hit.edu.cn

2015年10月16日

主要内容

1. 主流软件形态：SaaS
2. SaaS的部署环境：云平台

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

1. 主流软件形态：SaaS

什么是SaaS？

- **SaaS (Software-as-a-service，软件即服务)**
  - 一种通过Internet提供软件的模式，用户不用再购买软件，而改用向提供商租用基于web的软件，来管理企业经营活动，且无需对软件进行维护，服务提供商负责软件的可用性(软件维护、可扩展性、灾难恢复等)管理与支持；
- **企业采用SaaS服务模式，就像使用自来水和电能一样方便，从而大幅度降低了组织中应用先进技术的门槛与风险。**
- **关键词：On-demand licensing and use**

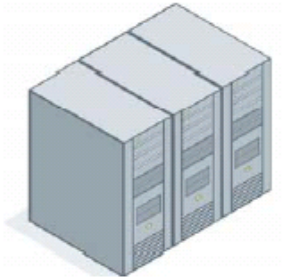
“自己造电厂”

VS

“订购电能”

软件架构的演化过程

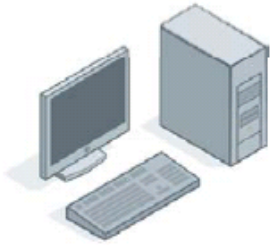
Mainframe



Mid 20th Century Platforms

IBM digital

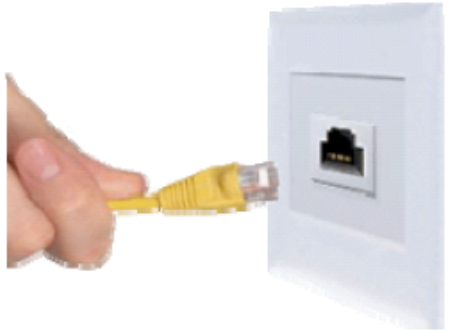
Client/Server On-Premise



Late 20th Century Platforms

ORACLE SIEBEL PeopleSoft SAP Microsoft

Software as a Service

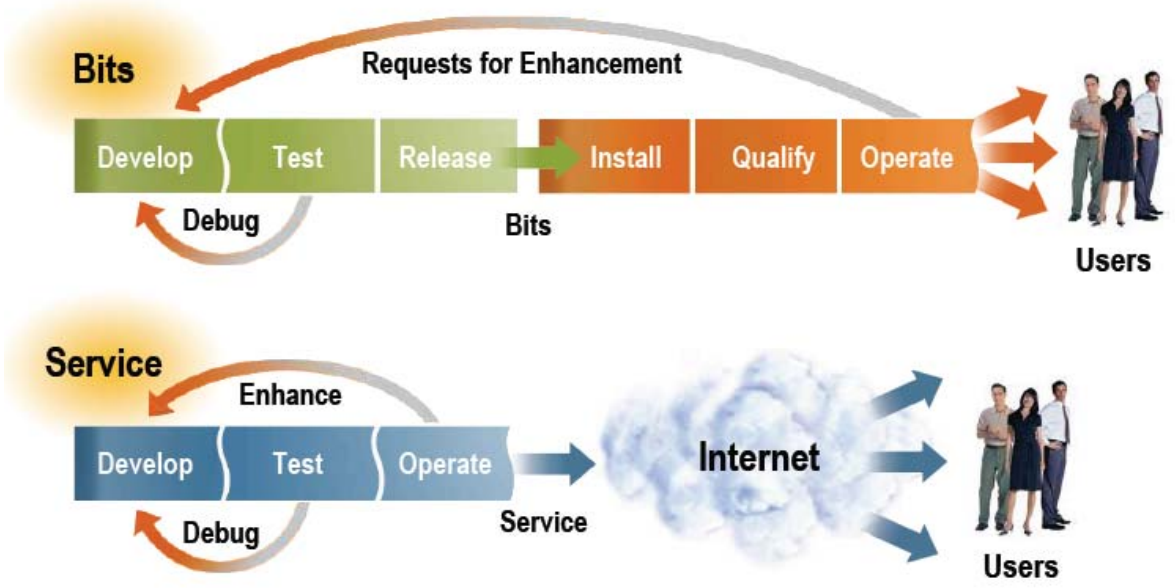


21st Century Platforms

Google ebay amazon.com salesforce.com Success On Demand.

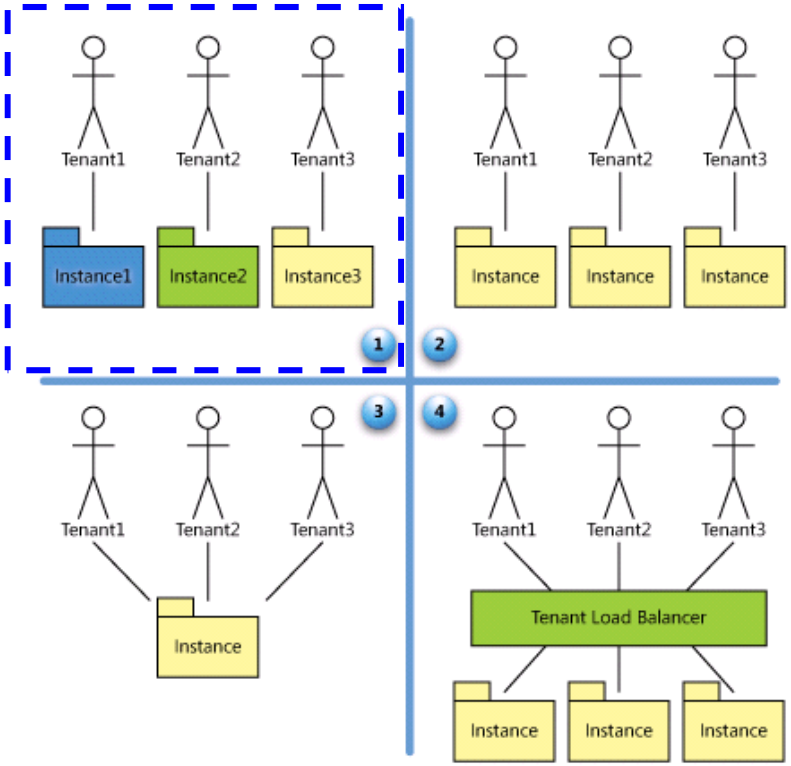
SaaS的特征

- **SaaS：本质上属于B/S结构，对B/S的扩展：**
  - 通过web来管理和使用软件；
  - 软件被集中式的部署与管理，统一升级和维护；
  - 单实例、多租户；
- **SaaS与传统B/S的本质区别：多组合共享Server和软件实例。**



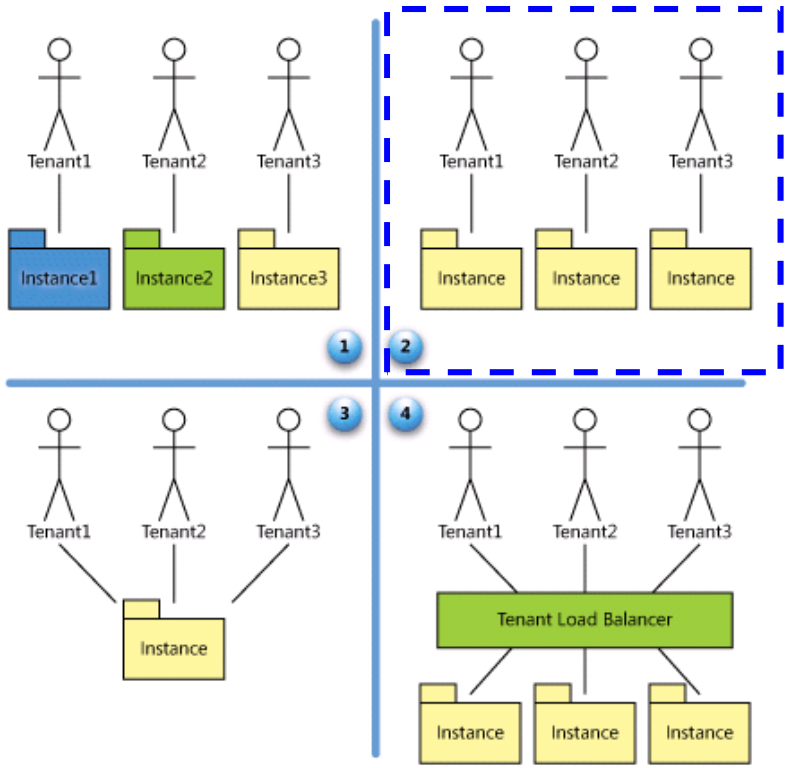
从package-based software到SaaS的四个阶段

- **Level 1 (Ad-Hoc/Custom) 定制开发**
  - 将传统的软件系统迁移为基于网络的应用；
  - 每个用户具有自己的一套独立系统(DB、app、code)，在提供商的硬件环境下运行自己的实例。



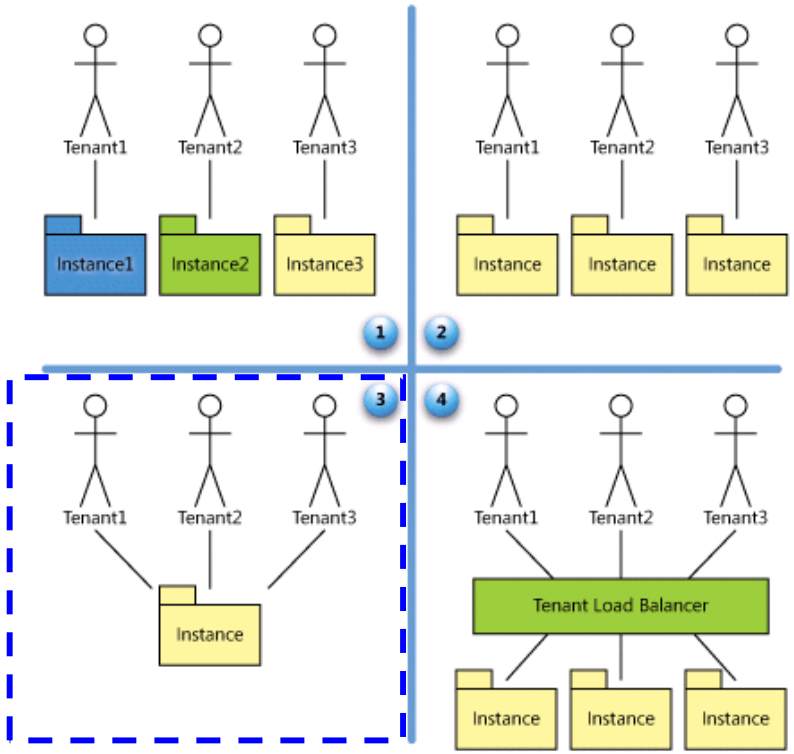
从package-based software到SaaS的四个阶段

- **Level 2 (Configurable) 可配置**
  - 通过配置，一套相同的软件系统可以适应不同顾客的需求；
  - 运行时，该系统为不同顾客产生不同的运行实例；



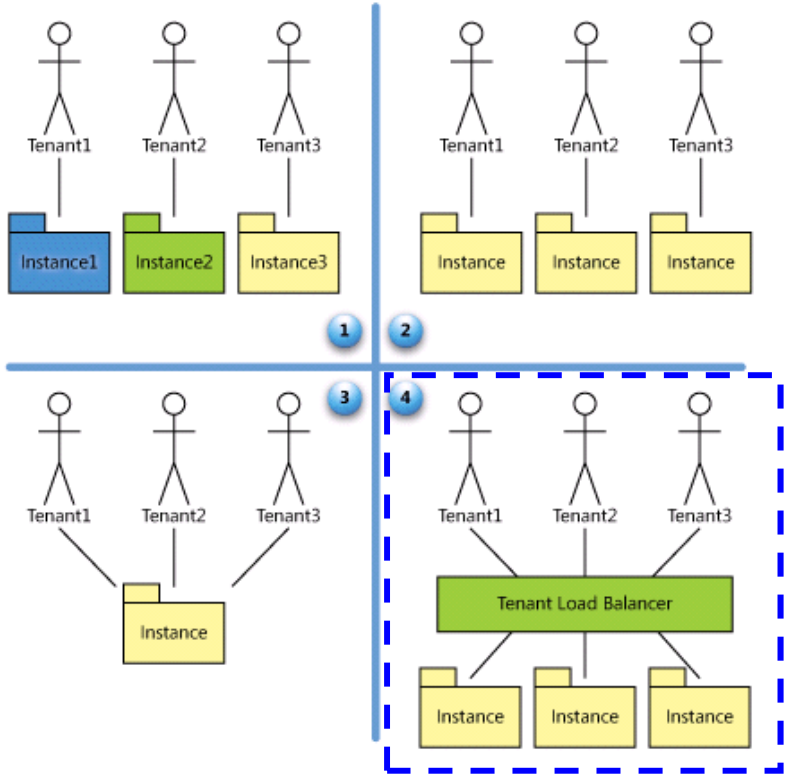
从package-based software到SaaS的四个阶段

- **Level 3 (Configurable, Multi-Tenant-Efficient) 高性能的多租户架构**
  - 在前一阶段的基础上增加了“多租户”特性；
  - 因而，多个顾客可以同时使用一个程序实例。
  - 优点：服务资源共享，利用效率高。

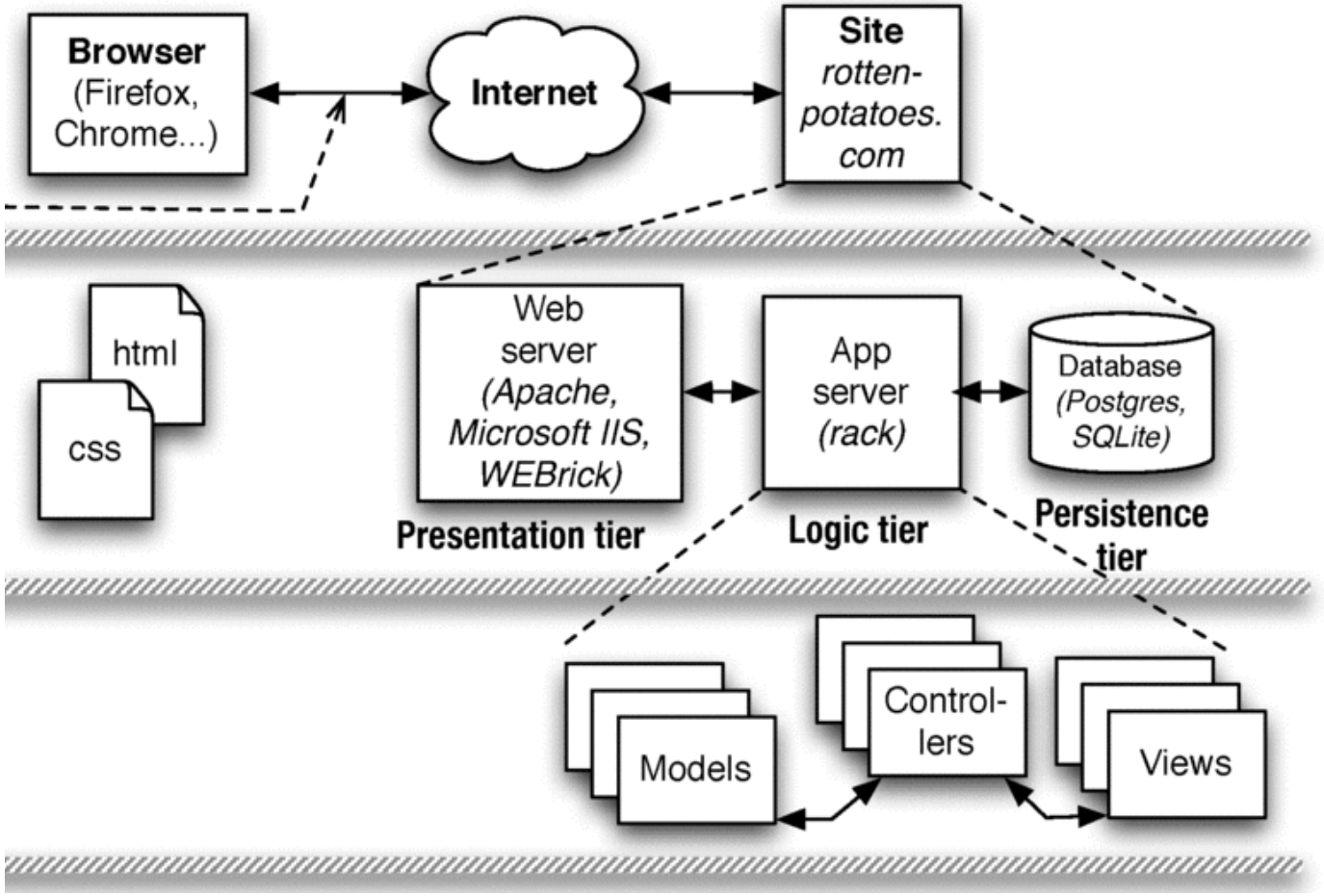


从package-based software到SaaS的四个阶段

- **Level 4 (Scalable, Configurable, Multi-Tenant-Efficient)**
  - 针对level 3中可能出现的资源伸缩性问题，增加了负载平衡功能。
  - 在多台服务器上部署多个instance，顾客请求被分配到不同实例上。
  - 提供商可根据需求大小，动态调整资源，而无需改变软件本身。



SasS的层次划分


Browser
(Firefox,
Chrome...)
Internet
Site
rotten-
potatoes.
com
html
css
Web
server
(Apache,
Microsoft IIS,
WEBrick)
App
server
(rack)
Database
(Postgres,
SQLite)
Presentation tier
Logic tier
Persistence
tier
Models
Control-
lers
Views

SasS的层次划分

表示层

1 接收用户输入的数据与指令、展示数据的处理结果；
2 本身并不维护数据，也不包含业务逻辑；

数据/指令

数据

控制层

1 统一维护一组对象；
2 根据界面层发来的指令，调度这些对象的行为；
3 除了调度之外，本身也可能包含业务逻辑(处理对象集合的逻辑)；
4 亦可直接访问DB；

逻辑层

对象(数据+业务逻辑)

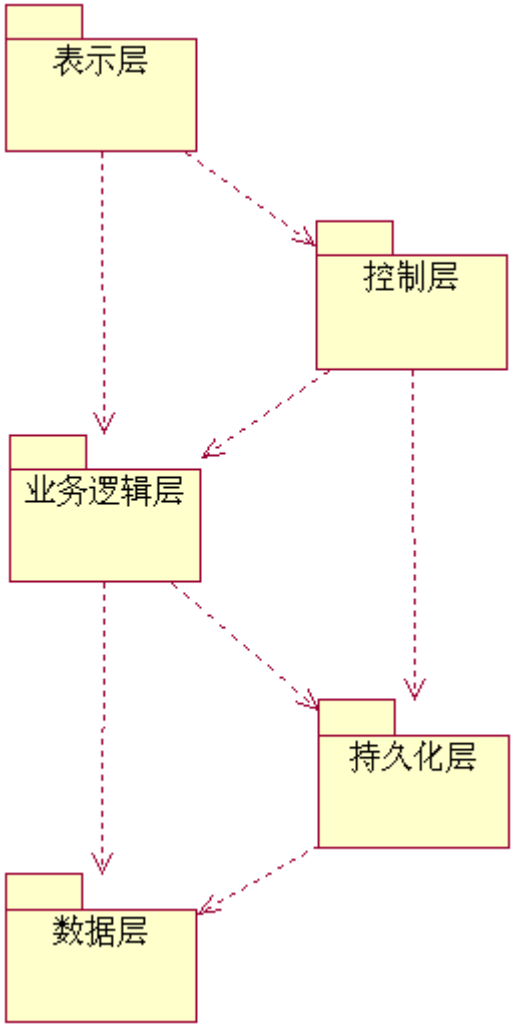
数据的持久化

持久化层

数据的持久存储(DB/file)

有关分层的补充说明

- 表示层、控制层、逻辑层、持久化层、数据层：这种划分是否一定有必要？——NO
- 这种划分的目的是为了“清晰分工”、“职责明确”，但同时也增加了系统运行时的性能代价；——为完成一项功能需要在多层之间多次调用
- 为简化起见，可将其中某些部分合并：
  - 边界类与控制类的合并：在用户界面中直接嵌入业务逻辑；
  - 控制类与实体类的合并：控制类中包含所有的业务逻辑，直接与持久化存储打交道。
  - 合三为一，不分层。
- 需根据系统的NFR需求，做出最佳的选择。

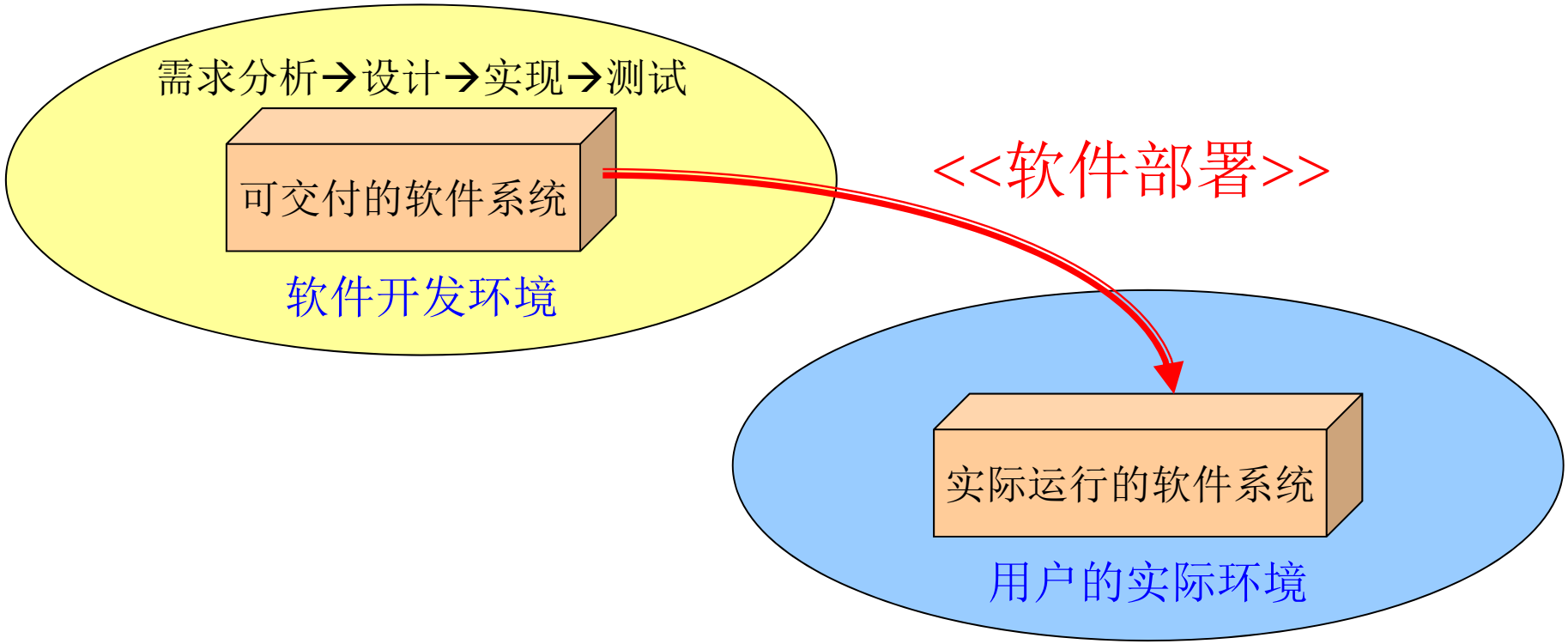


哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2. SaaS的部署环境：云平台

何谓“软件部署”

- 软件部署与实施(Software Deployment & Implementation)：将系统设计方案与软件系统转换成实际运行系统的全过程。

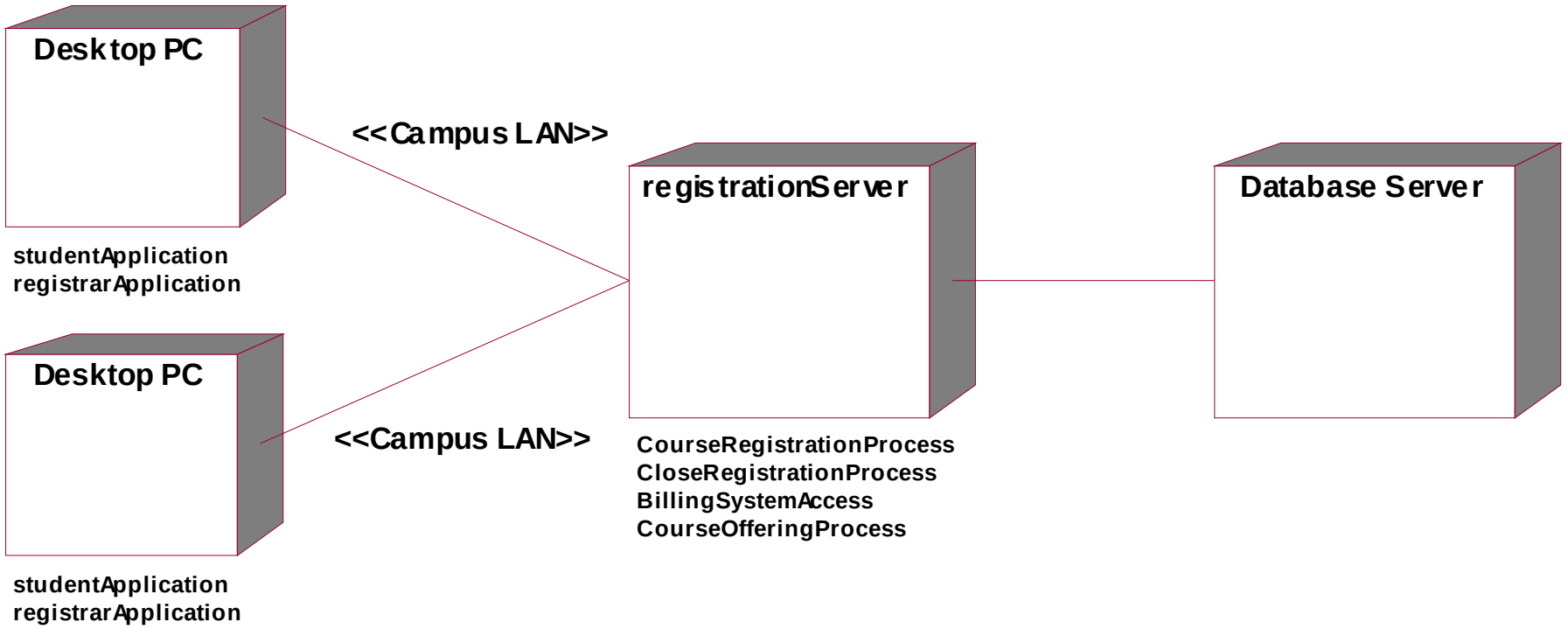


软件部署模型(Deployment Model)

- 为系统选择硬件配置和运行平台；
- 将类、包、子系统分配到各个硬件节点上。

- 系统通常使用分布式的多台硬件设备，通过UML的部署图(deployment diagram)来描述；
  - 部署图反映了系统中软件和硬件的物理架构，表示系统运行时的处理节点以及节点中对象/子系统的分布与配置。
  - 部署对系统的性能和复杂度具有较大影响，需要在设计初期就要完成。
- 描述系统中各硬件之间的物理通讯关系；
- 描述各软件实体被配置到哪个具体硬件上、这些软件实体之间的物理通讯关系；

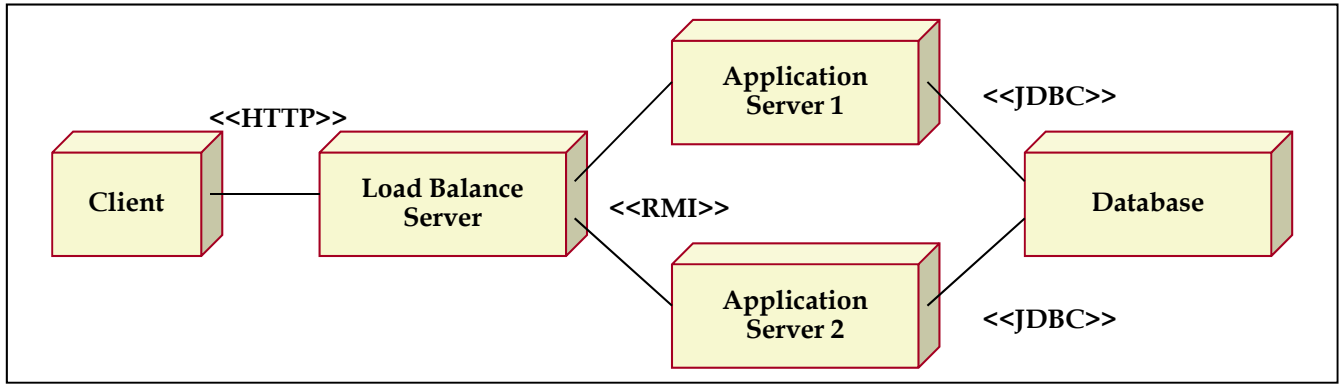
部署子系统

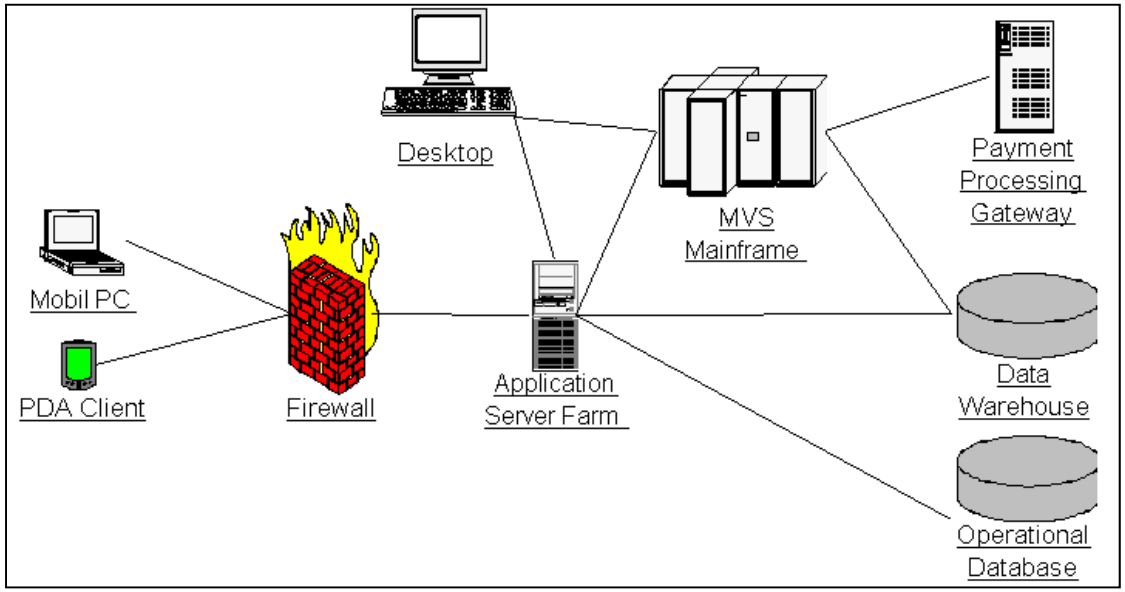

Desktop PC
studentApplication
registrarApplication
<<Campus LAN>>
registrationServer
Database Server
Desktop PC
studentApplication
registrarApplication
<<Campus LAN>>
CourseRegistrationProcess
CloseRegistrationProcess
BillingSystemAccess
CourseOfferingProcess

关于部署图

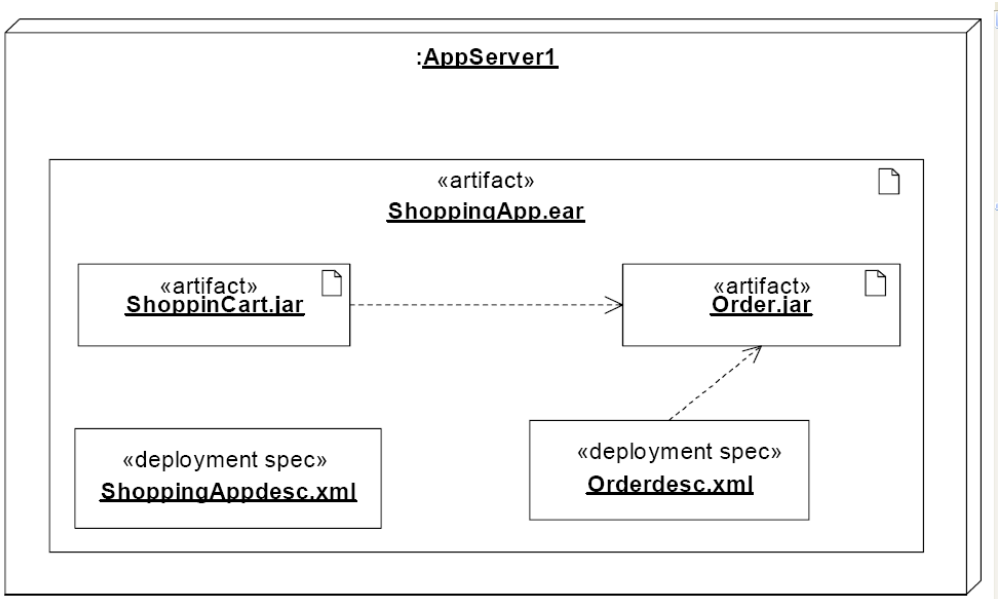
- **部署图(deployment diagram)：**
  - 节点(node)：一组运行资源，如计算机、设备或存储器等。每个节点用一个立方体来表示，
  - 节点的命名：client、Application Server、Database Server、Mainframe等较通用的名字；
  - 节点立方体之间的连接表示这些节点之间的通信关系，通常有以下类型：异步、同步；HTTP、SOAP；JDBC、ODBC；RMI、RPC；等等；
- **部署图在两个层面的作用：**
  - High-level：描述系统中各硬件之间的物理通讯关系；
  - Low-level：描述各软件实体被配置到哪个具体硬件上、这些软件实体之间的物理通讯关系；

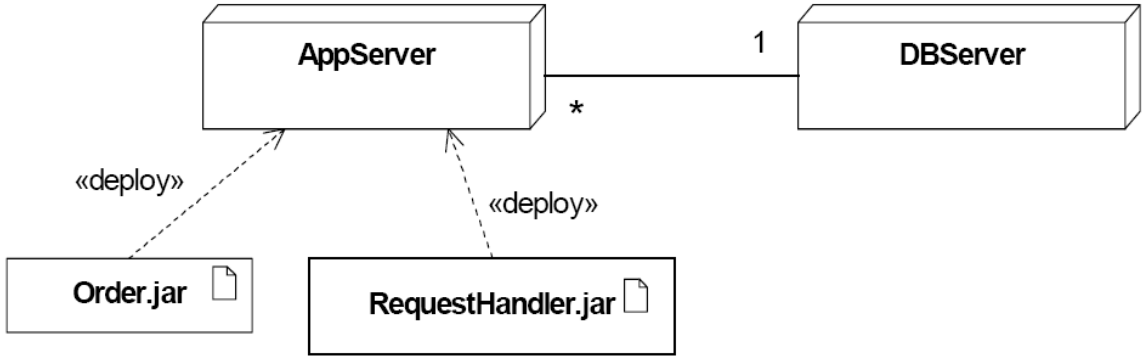
High-level Deployment Diagram





Low-level Deployment Diagram





绘制部署图(deployment diagram)

- **确定“节点(node)”：**
  - 标识系统中的硬件设备，包括大型主机、服务器、前端机、网络设备、输入/输出设备等。
  - 一个处理机(processor)是一个节点，它具有处理功能，能够执行一个组件；
  - 一个设备(device)也是一个节点，它没有处理功能，但它是系统和外部世界的接口。
- **对节点加上必要的“构造型(stereotype)”**
  - 使用UML的标准构造型或自定义新的构造型，说明节点的性质。
- **确定“连接(connection)”**
  - 把系统的包、类、子系统、数据库等内容分配到节点上，并确定节点与节点之间、节点与内容之间、内容与内容之间的联系，以及它们的性质。
- **绘制配置图(deployment diagram)**

架构设计思路小结

- 逻辑架构：只考虑如何分层、每个层次中的模块、层次内模块的关系、层次间模块的关系。主要是给开发者提供指南。
- 物理架构：考虑的是实际硬件/网络环境，以及如何将逻辑架构映射到硬件/网络环境上去。主要是给实施人员提供指南。
  - 通常，逻辑分层可以1:1映射到物理分层上；
  - 某些时候，多个逻辑层次可以部署在同一个物理层次上(n:1)；
  - 某些时候，同一个逻辑层次可以拆分为多个物理设施上(1:n)；
- 物理架构的设计思路：
  - 从逻辑架构入手，分别考虑每个逻辑层次在物理环境下是如何实现的；
  - 从简单到复杂，考虑每项设计决策对物理设施的要求，逐渐扩充物理架构。

将SaaS部署到哪里？

- 三种选择：
  - 本机(主要用于开发环境)
  - 自己搭建服务器(组织内部使用)
  - 公共的服务器——云

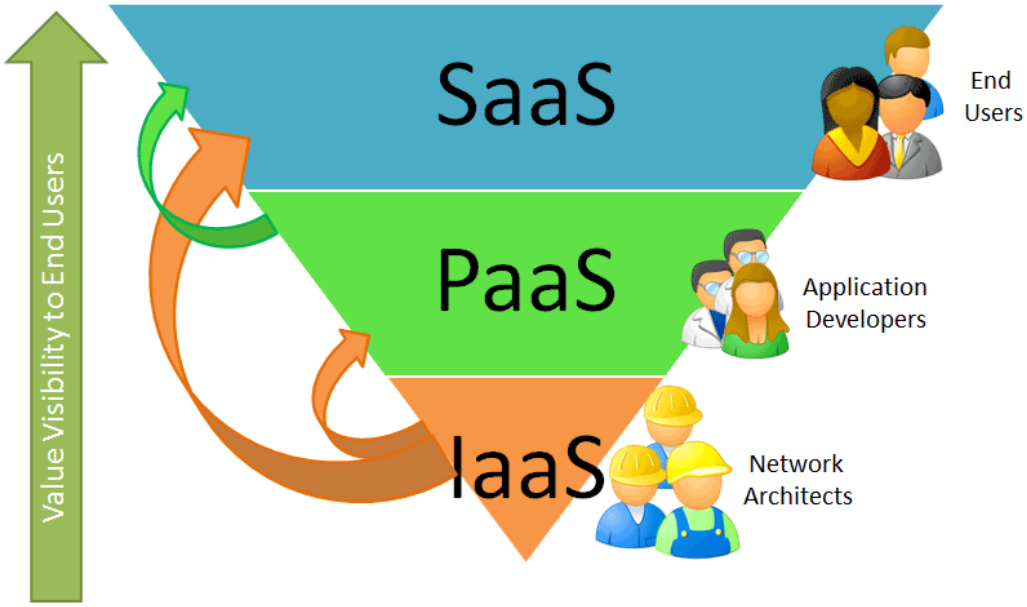
什么是Cloud?

- 这是一种新的计算方式和共享基础架构的方法，IT相关的计算能力被作为“服务”，通过Internet向外部客户提供，但客户不需了解这些计算能力的物理来源及其分布。

- 目标：使IT计算能力(存储和计算)可以向电能一样提供给客户。

Cloud所能提供的三种典型服务

- **IaaS (Infrastructure as a Service，基础架构即服务)**
  - 通过互联网提供了数据中心、基础架构硬件，可以提供服务器、操作系统、磁盘存储、数据库和/或信息资源。
  - Amazon EC2
- **Paas (Platform as a Service，平台即服务)**
  - 提供了软件基础架构，软件开发者可以在这个基础架构之上建设新的应用，或者扩展已有的应用。
  - Salesforce.com的Force.com、Google的App Engine和微软的Azure、新浪的SAE、百度的BAE
- **SaaS (Software as a Service，软件即服务)**
  - Salesforce.com、NetSuite、Google的Gmail/Docs



基础设施即服务(IaaS)

- 简单的说，**IaaS**可看作物理服务器(裸机，**CPU+**内存**+**磁盘**)**的虚拟化；
- 用户可在上面安装操作系统、运行环境、装载数据，再在上面部署应用系统。
- 代表：
  - Amazon EC2
  - Google Compute Engine (GCE)
  - OpenStack
  - VMWare
  - Eucalyptus

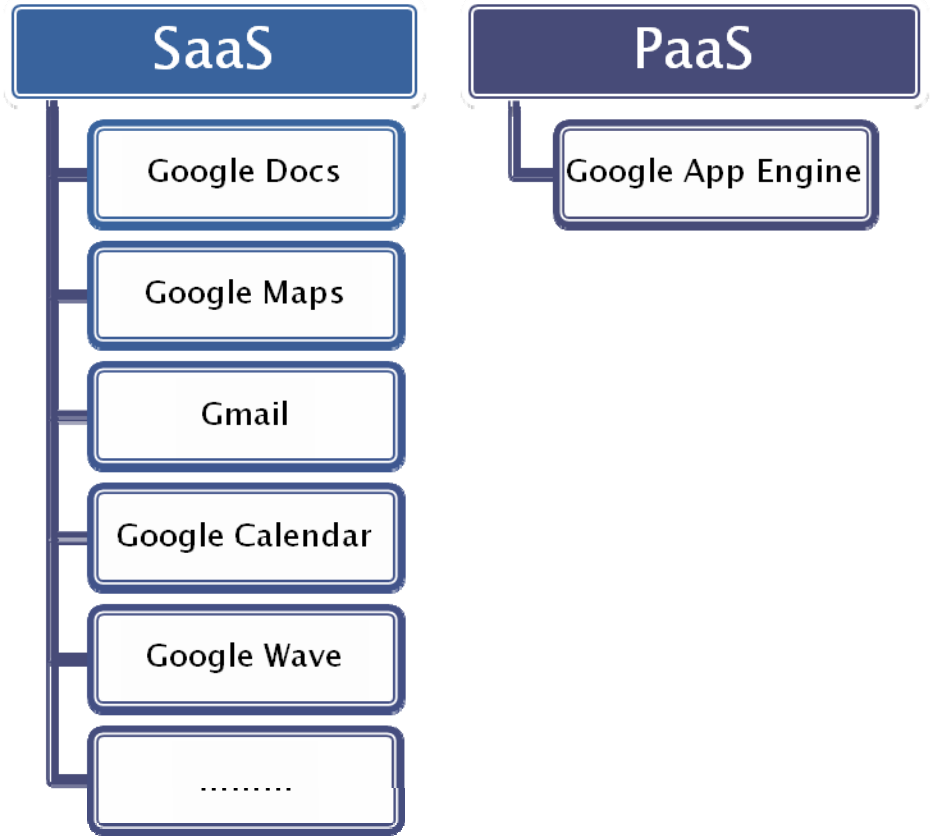
- 哈工大的云空间 **http://s.hit.edu.cn/**

平台即服务(PaaS)

- 在**IaaS**基础上，有了完整的运行环境和基础服务支持(例如**OS**、**DB**、应用服务器、**MVC**、**Message**等)；
- 将中间件环境作为了服务，向用户提供；
- 按照平台要求将程序部署到上面去。
- 代表：
  - Google App Engine (GAE)
  - Windows Azure
  - Sina App Engine (SAE)
  - Baidu App Engine (BAE)
  - Heroku

Example: Google

- Google云计算应用实例



Google 文件

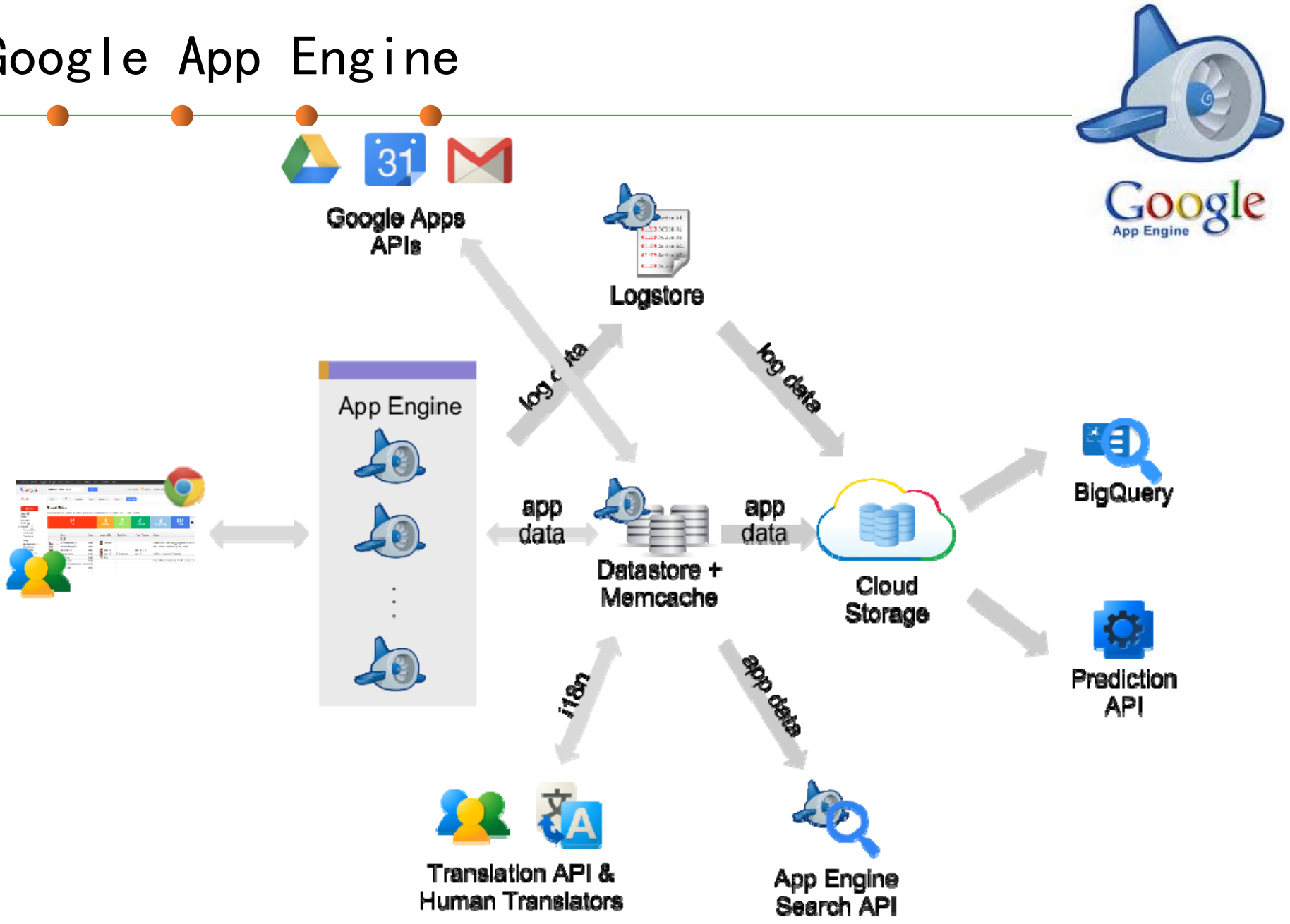
Google 地图 谷歌 中国

Google 日历

Gmail by Google BETA

Google app engine

Google App Engine



SAE作为PaaS


English | 活动 | 注册 | 登录
SinaAppEngine
已关注
输入想要搜索的关键字
搜索
首页
架构
成功案例
资费
文档中心
服务列表
开发者社区
新手指南
分布式WEB服务的开发、运行平台
负载均衡
Load Balancer
Web服务器
Web Servers
缓存
Cache
数据库
Database
安全
Security
任务调度
Task Schedulers
存储服务
Storage
扩展服务
Extension

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

结束

2015年10月16日