服务计算概述
Introduce to Service Computing

中山大学
数据科学与计算学院

潘茂林
panml@mail.sysu.edu.cn

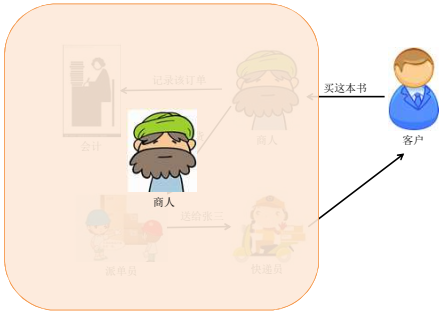
大纲

- 理解面向服务
- 服务计算-分布式计算的一个分支
- 服务计算与IBM
  - 面向服务的架构（SOA）
- 服务计算遇上云计算
  - NetFlix OSS 与 微服务架构
  - 面向服务的编程
- Go语言简介
  - 以软件工程为目的的语言设计
  - 系统工程师的语言
  - 云开发首选语言

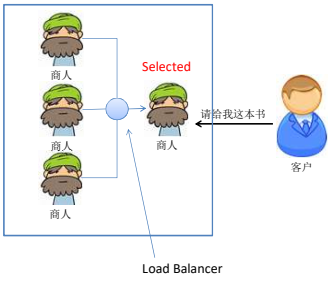
面向服务 - 客户与服务者

这里：
- 服务是一组能力（Capabilities）的集合
- 服务者（Server）是拥有能力的实体
- 客户是（Client’）是使用能力的实体

面向服务 - Agnostic Logic



面向服务 - 可伸缩（Scaling）



面向服务 - 服务注册与发现

面向服务的优点

- 简单，仅考虑能力描述
  - 易于标准化，例如制定会计的服务标准
- Agnostic交付，即服务实现是黑盒的
  - 服务内部升级变化，不影响客户使用
- 可组合，灵活
  - 通过服务编排，推动业务创新，易于实现自动化
- 可扩展、可优化
  - 通过服务组合或组合优化提升服务容量和水平
  - 通过伸缩适应服务容量
- 高可靠
  - 使用冗余的服务资源组成特定的高可靠结构
- 商业创新
  - 通过服务市场交易，创新软件使用模式（相对于 On-Premises）

服务与服务计算的定义

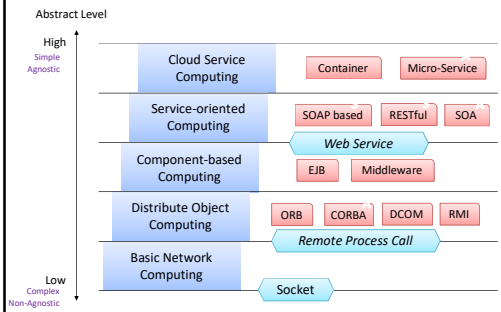
- 服务（计算机科学定义）

  From a general perspective, a *service* is a software program that makes its **functionality** available via a published API that is part of a *service contract*.
- 服务计算

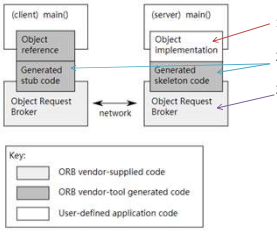
  Services Computing has become a cross-discipline that covers the science and technology of bridging the gap between Business Services and IT Services. The underlying breaking technology suite includes Web services and service-oriented architecture (SOA), cloud computing, business consulting methodology and utilities, business process modeling, transformation and integration. This scope of Services Computing covers the whole lifecycle of services innovation research that includes ***business componentization, services modeling, services creation, services realization, services annotation, services deployment, services discovery, services composition, services delivery, service-to-service collaboration, services monitoring, services optimization***, as well as ***services management***. The goal of Services Computing is to enable IT services and computing technology to perform business services more efficiently and effectively.

  IEEE Technical Committee on Services Computing

分布式计算技术发展历程



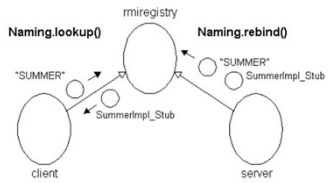
相关技术-Object Request Broker



1. 用户编写对象实现
2. 生成服务端、客户端程序
3. 编译，连接ORB库

https://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture

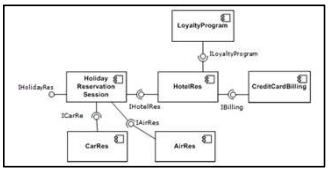
相关技术 - RMI Registry



服务定位技术

1. 服务端注册
2. 客户端查询服务位置
3. 客户端访问服务

相关技术 - Component-based Software



简单的例子，一个假想的假日酒店预订系统的组件设计，用UML2.0表示

一个软件系统，通过研究它们业务之间的关系，分解为若干部件，部件与部件之间用接口连接。

IBM 简史

- 1911年6月16日，计算制表计时公司宣告成立
- 1923年，发明了打孔卡数据处理技术
- 1934年，推出机电驱动的405型字母会计运算机

美国1935年颁布了《社会保障法》，出于会计计算的要求，政府、企业项目无数

- 1948年，推出可选序列电子计算器(SSEC)，免费给客户使用
- 1962年，7090型大型计算机，用于美国航空公司的实时订票系统
- 1964年，S/360大型机大获成功，银行、保险、宇航局成为客户
- 1975年，IBM推出首款便携式计算机
- 1988年，IBM发布AS/400服务器产品线，主推中小企业

http://tech.sina.com.cn/it/2011-06-16/00555652656.shtml

郭士纳新政：大力投资软件和服务

- 1969年，美国司法部发起反垄断诉讼，官司打了13年
- 1993年，IBM因盲目扩张陷入亏损，并首次外聘CEO
- 郭士纳新政：
  - 剥离如DRAM、网络、个人打印机和硬盘等低利润业务
  - 收购莲花软件和普华永道，大力发展软件和服务业
- 1997年，IBM研发的超级计算机“深蓝”在一场国际象棋比赛中击败世界冠军卡斯帕罗夫，创造历史。
- 2005年，出售PC业务，筹集100多亿美元在信息管理、商业智能和数据分析等领域，收购包括FileNet、Cognos、SPSS、iLog和Netezza在内的多家公司，以推动当时的“智慧地球”战略。
- 2011年，IBM花费4年时间和无数资金研制“沃森”超级计算机，并让它参加智力问答电视节目，推动了人工智能技术发展。

郭士纳新政：大力投资软件和服务

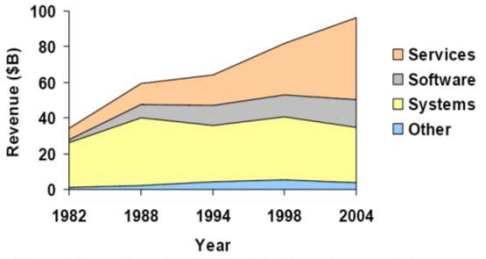
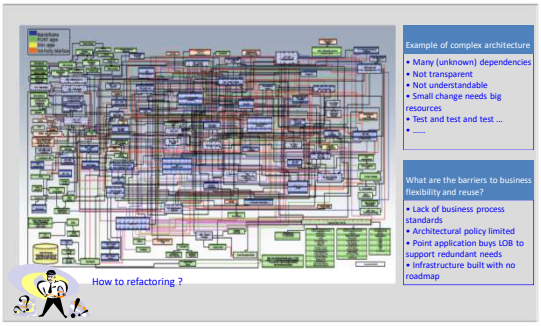
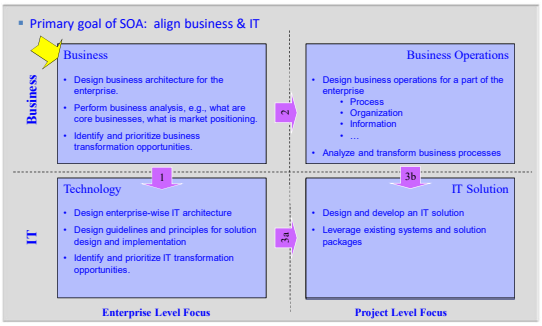


Figure 1: Increase in services revenue at IBM (Source: IBM Annual Reports).

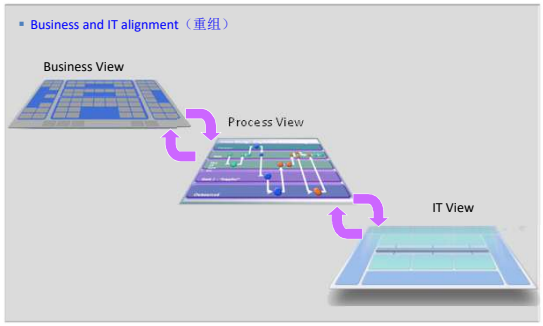
经济全球化时代IBM面临的困境



SOA 目标-业务与技术整合

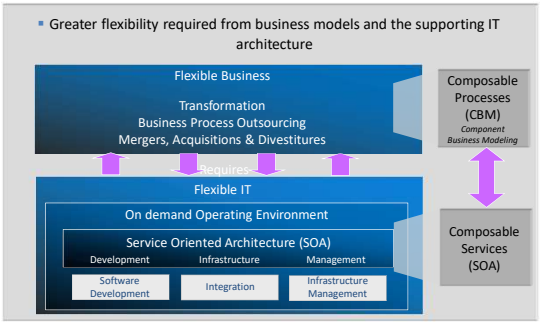


SOA 目标 - 支撑业务流程再造

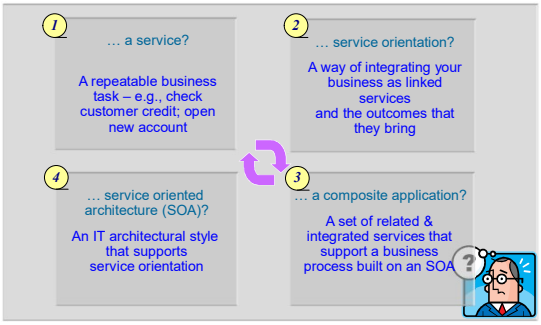


SOA目标 - 灵活的业务与IT结构

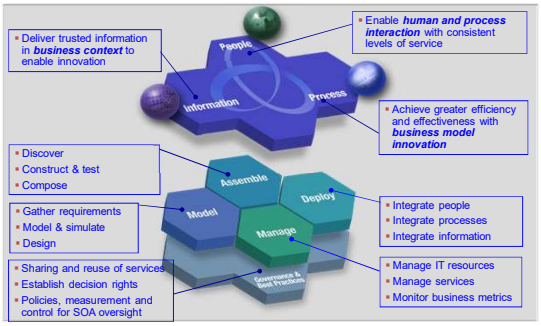
- Greater flexibility required from business models and the supporting IT architecture



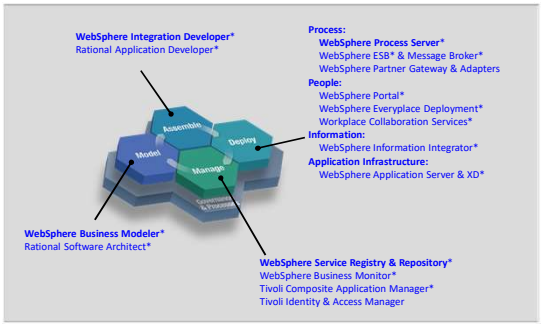
面向服务的思考过程



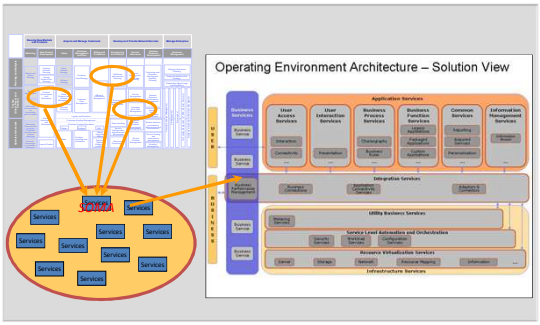
SOA生命周期 - 过程模型



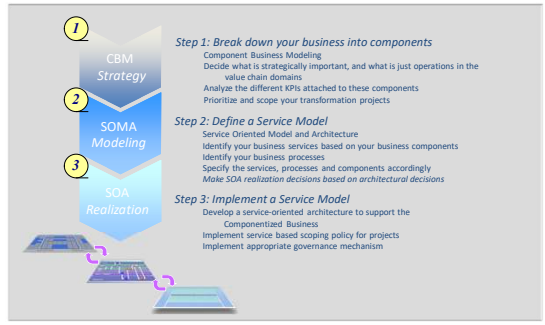
SOA生命周期 - 工具与产品



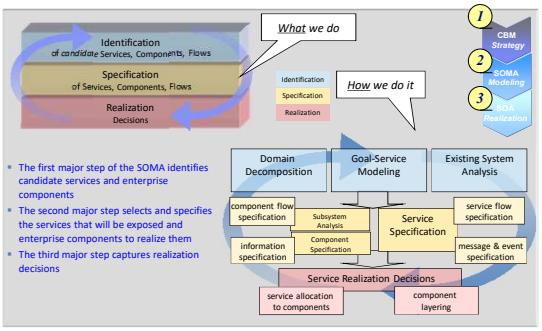
Service-Oriented Modeling and Architecture （SOMA）



Service-Oriented Modeling and Architecture （SOMA）

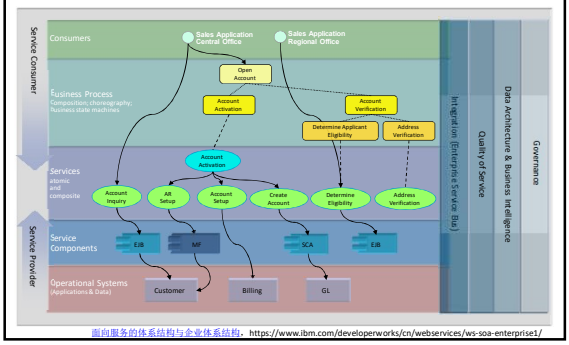


Service-Oriented Modeling and Architecture （SOMA）



- The first major step of the SOMA identifies candidate services and enterprise components
- The second major step selects and specifies the services that will be exposed and enterprise components to realize them
- The third major step captures realization decisions

SOA 解决方案堆栈

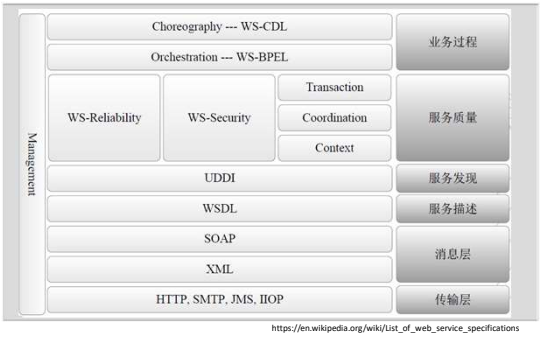


面向服务的体系结构与企业体系结构，https://www.ibm.com/developerworks/cn/webservices/ws-soa-enterprise1/

面向服务的架构（SOA）

- CBDI 的定义：
  - 一组策略、实践和框架，支持将应用程序功能作为一组服务在与能够调用、发布和发现的服务使用者相关的粒度发布；这组服务是使用接口的单一标准形式从实现抽象出来的。
- Gartner 的定义：
  - 面向服务的体系结构是一种客户机/服务器软件设计方法，其中的应用程序由软件服务和软件服务使用者（也称为客户机或服务请求方）组成。SOA 与更为通用的客户机/服务器模型不同，其定义强调软件组件间的松散偶合及对独立接口的使用。
- IBM 定义：
  - “面向服务的体系结构是一种用于根据需要对资源进行关联的企业级 IT 体系结构。这些资源被表示为与业务一致的服务，这些服务可以参与和包含到价值网。企业或业务线中，以满足业务需求。SOA 应用程序的主要结构化元素是服务，而不是子系统、系统或组件。”
  - 业务角度的定义：
    - 业务希望向其客户及合作伙伴或组织的其他部分公开的服务集。
  - 体系结构角度的定义：
    - 需要服务提供者、请求方和服务描述的体系结构风格。
    - 一组体系结构原则、模式和标准，以处理各种特征，如成熟度、封装、松散偶合、关注分离、重用、可组合性和单一实现。
  - 实现角度的定义：
    - 一种包括诸如 Web 服务等标准、工具和技术的编程模型。

SOAP-based Web Service 标准



https://en.wikipedia.org/wiki/List_of_web_service_specifications

IBM SOA 的核心技术

- 服务发现与集成（Universal Description Discovery and Integration，UUDI）：服务仓库。服务在UDDI中心注册、用户查找并使用。

  全球服务注册中心！！！
- 业务流程执行语言（Business Process Execution Language，BPEL）：服务组合语言。通过组合服务，生成新的服务。

  通过服务组合生产新应用！！！
- 企业服务总线（Enterprise Service Bus，ESB）：分布式中间件。这些注册的、组合的服务，包括传统消息、邮件API的都挂在一条总线上。

  随用随取，随时修改！！！

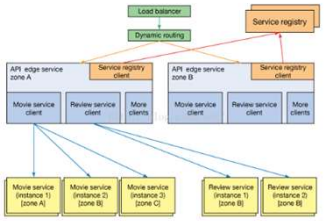
RESTful Web Service – 程序员的选择

- 2000年，Roy Fielding 在毕业论文提出 REST 概念
- Restful web service
  - Representational state transfer (REST) or RESTful Web services are one way of providing **interoperability** between computer systems on the **Internet**.
  - REST-compliant Web services allow requesting systems to access and manipulate **textual** representations of **Web resources** using a **uniform** and predefined set of **stateless** operations.
- REST 优势 – 更简单，高效，易于理解
  - 使用 JSON，XML，TEXT 传输对象与数据
  - 与普通 Web 程序访问 Internet 文档方法兼容，无状态
  - 资源面向的数据模型
  - 云环境服务的事实标准

https://en.wikipedia.org/wiki/Representational_state_transfer

Netflix OSS 与 云应用

- 2009年，NetFlix 在 Amazon AWS 建立了高性能、高可用、高可靠、可伸缩、易于维护的视频服务应用。



NETFLIX OSS

Netflix OSS 与 云应用

- AWS ELB（弹性负载均衡）
- Eureka（服务注册与发现）
  - AWS Service registry for resilient mid-tier load balancing and failover.
- Archaius（配置管理中心）
  - 与云平台配合缩放计算资源，安装与管理分布式的服务
- Ribbon（远程过程调用/含负载均衡策略）
  - Ribbon is a Inter Process Communication (remote procedure calls) library with built in software load balancers.
- Hystrix（断路器）
  - 抑制短暂的请求过载，隔离潜在的故障服务
- Netflix Zuul（API 网关服务）
  - Zuul is an edge service that provides dynamic routing, monitoring, resiliency, security, and more.
- Service Instances（服务实例）
- 除了AWS ELB外，以上都是以REST为基础的服务/管理服务

云应用是由一组管理服务和业务服务组合而成的应用？

Netflix OSS 与 云应用

Netflix微服务化支撑业务千倍增长

Netflix已经全部迁移到云端，http://www.infoq.com/cn/news/2016/02/Netflix-move-cloud

单体架构 pk 微服务架构

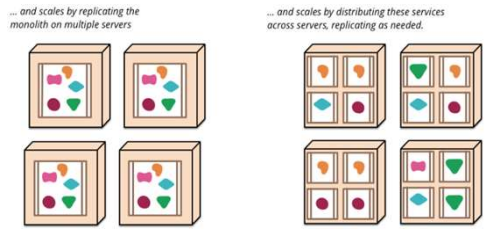


Figure 1: Monoliths and Microservices

https://martinfowler.com/articles/microservices.html

微服务架构 - 九大特征

- 特性一：“组件化”与“多服务”
- 特性二：围绕“业务功能”组织团队
- 特性三：“做产品”而不是“做项目”
- 特性四：“智能端点”与“傻瓜管道”
- 特性五：“去中心化”地治理技术
- 特性六：“去中心化”地管理数据
- 特性七：“基础设施”自动化
- 特性八：“容错”设计
- 特性九：“演进式”设计

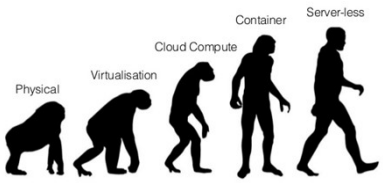
https://martinfowler.com/articles/microservices.html

多租户-虚拟化（隔离）与共享

	处理器	内存	操作系统	进程组	主机名	网卡	文件系统	桌面	其他IO设备
进程（IPC）	隔	隔	共	-	共	共	共	-	共
多用户	隔	隔	共	隔	共	共	隔	隔	共
虚拟机	隔	隔	隔	隔	隔	隔	隔	隔	隔
App引擎（沙箱）	隔	隔	-	-	隔	-	隔	-	-
容器	隔	隔	共	隔	隔	隔	隔	-	-
容器集群	隔	隔	共	隔	隔	隔	隔	-	-
Serverless*（Func-aaS）	-	-	-	-	隔	-	隔	-	-

多租户-虚拟化与服务

买房 >>>割单间 >>>租房 >>>合租 >>>住酒店



物理服务器→ 虚拟机 →云服务器→容器服务→函数服务

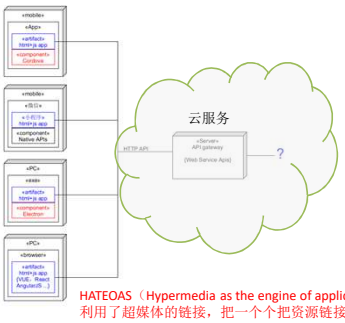
注意：软件工程技术不是赶时髦，每种技术都有自己合适的**特定的应用场景**

Serverless 应用开发指南，http://serverless.ink/

软件工程 - 面向服务

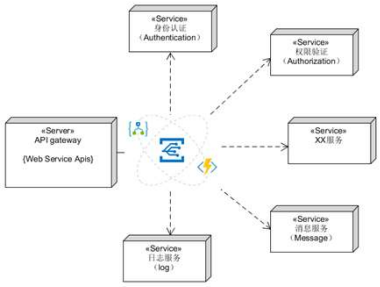
- 面向服务的业务分析
  - 基于容量规划的业务过程分解
- 面向服务的软件设计
  - 面向资源的设计
  - 软件架构 - 微服务架构
  - 服务计算架构模式
  - 服务编排与设计工具
- 面向服务的编程
  - 服务的编程与开发工具
  - 服务安全与多版本管理
  - 服务测试与部署的自动化（CI/CD）
- 服务的运维
  - 服务运行环境管理
  - 基于容量的自适应技术
  - 服务质量优化与监控

面向服务编程 – 统一富客户端

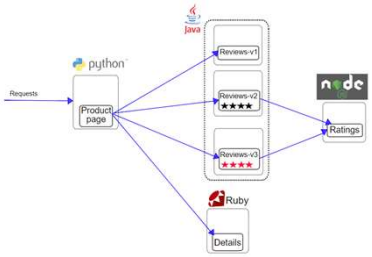


HATEOAS（Hypermedia as the engine of application state）利用了超媒体的链接，把一个个把资源链接起来。

面向服务编程 –依赖第三方服务



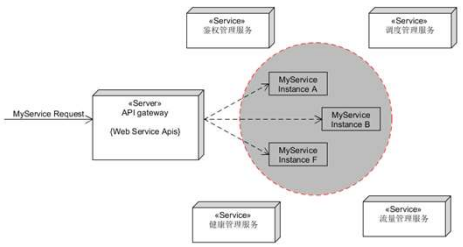
面向服务编程 –依赖第三方服务



案例：Bookinfo 是一个异构应用，几个微服务（其中 Reviews是多版本的）组合，且是由不同的语言编写的。

http://blog.daocloud.io/istio-canary-deployment/

面向服务编程 – 服务自治与幂等性



幂等性，简单的说就是任何服务实例执行请求，其输出是一样的。
这个服务编程约束是强制性的！

服务的基本概念

- 服务（Service）
  - 描述特定业务领域事务的功能集合，如谷歌地理信息服务
- 服务实例（Service Instance，有时简称 Service）
  - 提供服务的实体或软件服务的进程
- 软件镜像（Mirror）
  - 安装就绪的软件系统状态的持久化数据。通过分配IT资源创建服务。例如容器的镜像
- 仓库（Repository）
  - 保存镜像及其版本信息的存储空间，如 Docker hub
- 服务注册中心（Service Registry ）
  - Key-Value的数据库，key是服务名，value是服务实例地址
- 统一资源定位符（Uniform Resource Identifier /URI）
  - 服务API的定位描述，例如：/服务名/资源路径?参数
- API网关（API Gateway）
  - 按一定的策略将服务请求映射到对应的服务的实例

Go在谷歌：以软件工程为目的的语言设计

- 产生的背景
  - 多核处理器、系统的网络化、大规模计算机集群和Web编程模型
  - 提升编译、开发效率
- Go语言定位
  - Go语言开发自Google，是一门支持**并发编程**和内存**垃圾回收**的**编译型静态类型**语言
  - 面向系统工程师
    - 编译快速，简洁、快速、安全
    - 符合系统工程师编程习惯
    - 合适团队协作
    - 跨平台
  - 不是一门在研究领域里具有突破性进展的语言，但它却是大型软件项目中软件工程方面的一个非常棒的工具

Go在谷歌：以软件工程为目的的语言设计
https://www.oschina.net/translate/go-at-google-language-design-in-the-service-of-software-engineering?lang=chs&#

系统工程师的工作

- 常见任务
  - 系统资源调度与管理算法
  - 系统管理，包括：监控、调度、部署
  - 系统管理 web 服务程序
- 常用语言（JavaPythonGo工程师）
  - c/c++
    - 优点：最贴近硬件的高级语言，程序执行效率的基准
    - 缺点：内存管理，指针的使用
  - Python
    - 优点：最常用的胶水语言，开发效率高
    - 缺点：不合适大规模团队开发，执行效率低下
  - Java
    - 优点：最合适大规模团队协作的跨平台语言。
    - 缺点：效率一般，规范性太强

Go语言：云计算首选

- 云服务、分布式应用管理系统开发
  - Docker, a set of tools for deploying Linux containers
  - Kubernetes container management system
  - InterPlanetary File System, a content-addressable, peer-to-peer hypermedia protocol
  - OpenShift, a cloud computing platform as a service by Red Hat
- 各种系统运维工具与企业级高性能 web 应用
  - Caddy, an open source HTTP/2 web server with automatic HTTPS capability
  - Juju, a service orchestration tool by Canonical, packagers of Ubuntu Linux
- 实用计算最佳语言（经济性，简单性）
  - 接近Python的开发效率，接近c语言的运行效率
  - 优化的多线程支持，更低的通讯成本
  - Web应用承载能力是Java的数倍，合适云上部署

https://en.wikipedia.org/wiki/Go_(programming_language)#Projects