同濟大學
TONGJI UNIVERSITY

利用云计算平台设计ERP物流管理服务

Design ERP logistics management services using cloud computing platforms

答辩人: 肖鹏飞　　学号: 1953072　　方向: 工业软件与云计算　　指导老师: 曾国荪

选题概述

目标

选题要求基于**容器化微服务架构**设计一个**ERP物流管理系统**，并将其部署在**云计算平台**上。

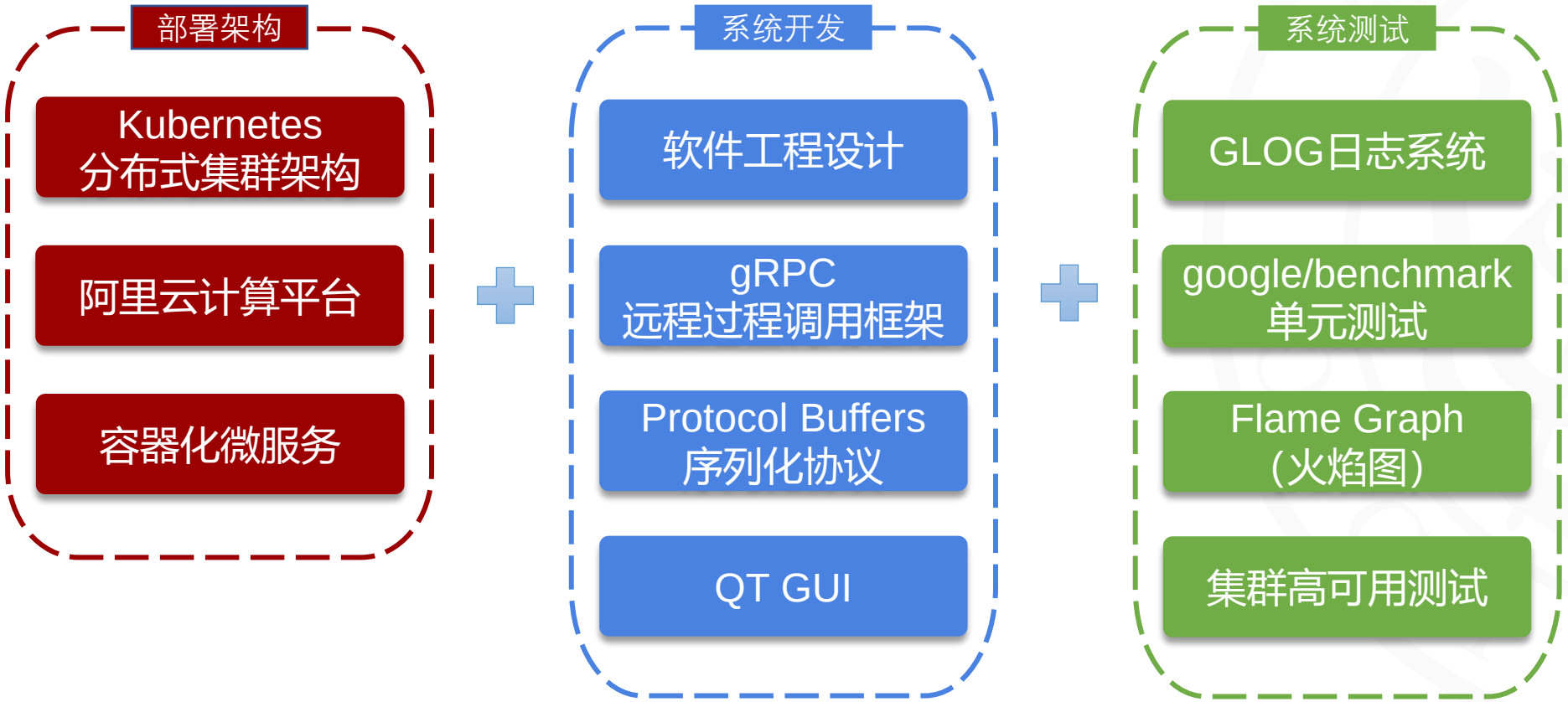
意义

- 云平台ERP**工业软件**有助于制造业转型升级
- 借用平台和算力实现**弹性资源调度**
- 利用云计算技术为传统需求赋能的必由之路

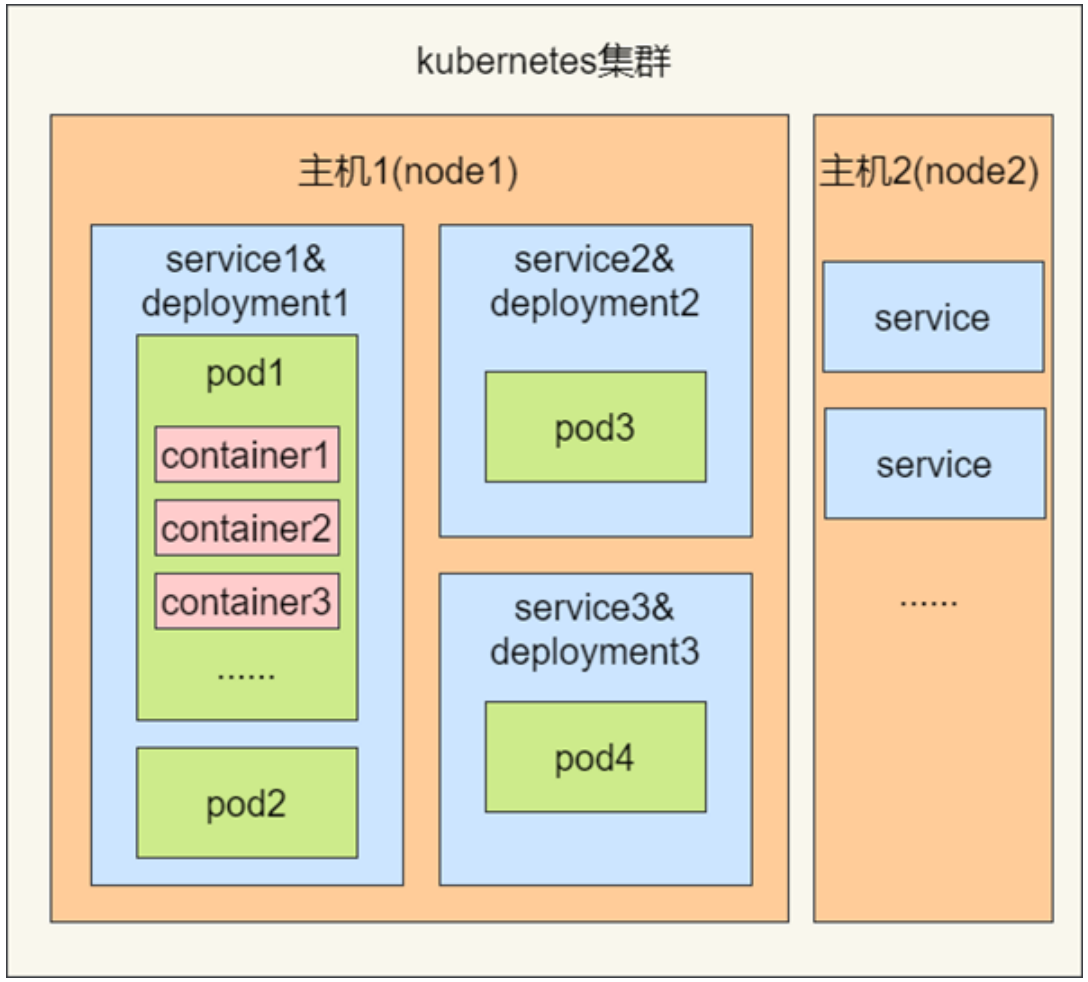
技术路线

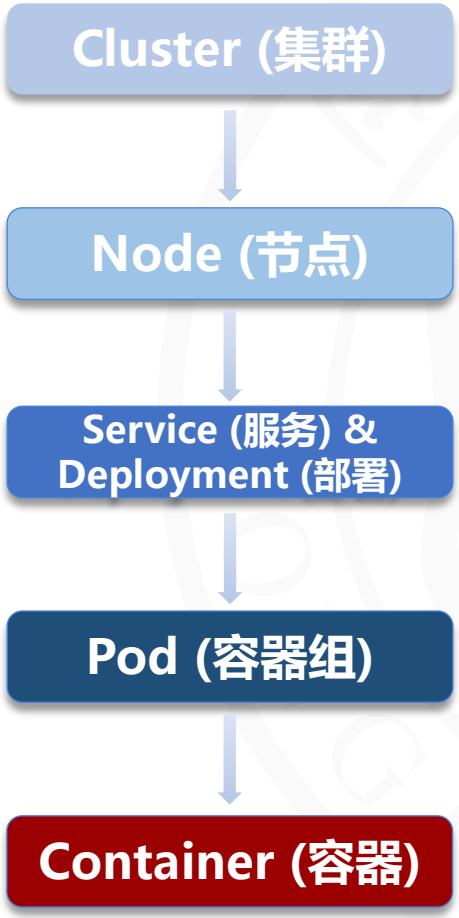
- Kubernetes分布式集群搭建
- 容器化微服务架构
- 阿里云计算平台服务
- 软件系统开发与功能验证

本文工作（技术栈）


部署架构
Kubernetes
分布式集群架构
阿里云计算平台
容器化微服务
系统开发
软件工程设计
gRPC
远程过程调用框架
Protocol Buffers
序列化协议
QT GUI
系统测试
GLOG日志系统
google/benchmark
单元测试
Flame Graph
（火焰图）
集群高可用测试

Kubernetes集群的自顶向下层级架构

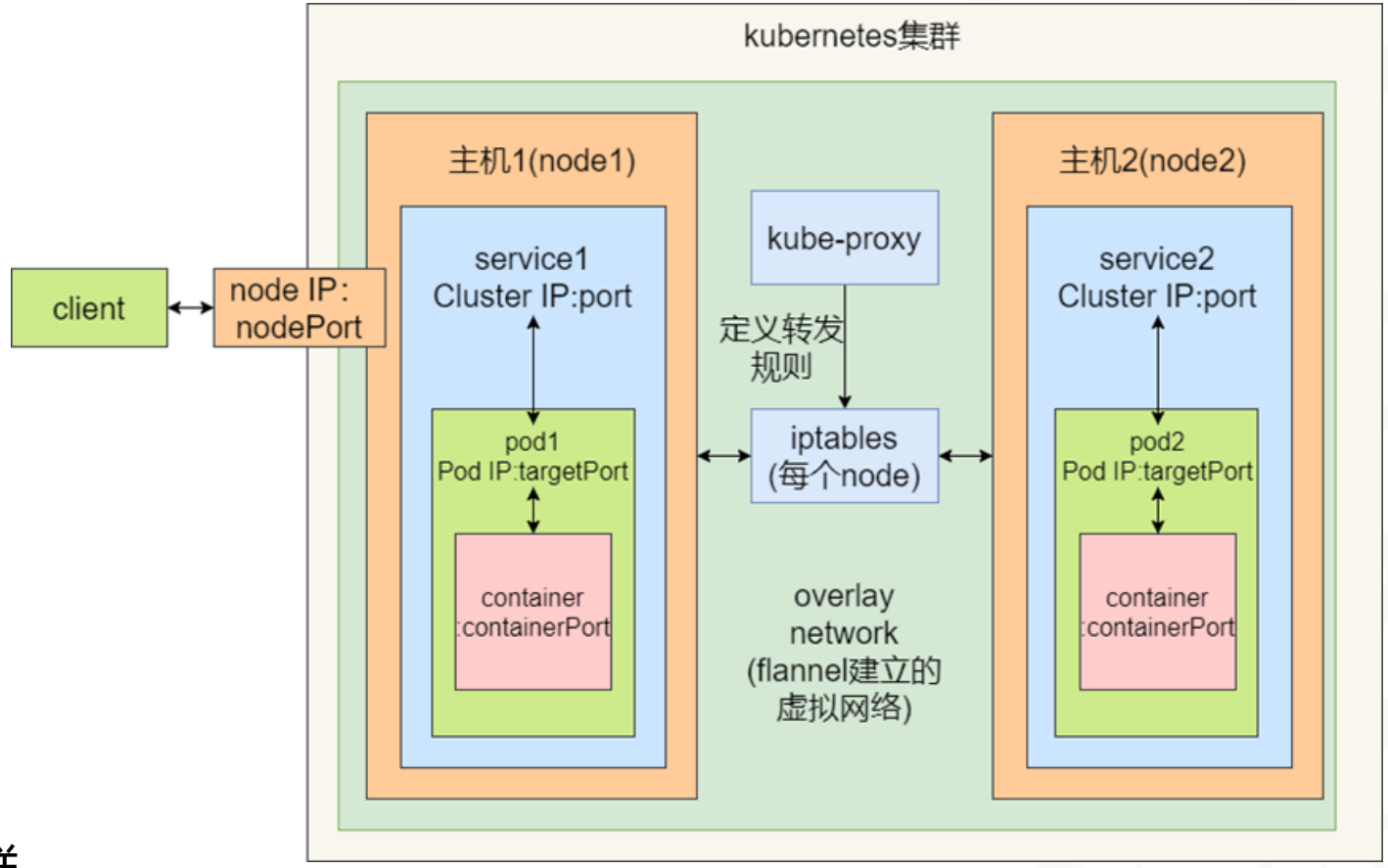




Kubernetes集群的通信模型

内部通信

Flannel网络插件，使用**kube-proxy**在Linux上使用**iptables**的方式来实现网络的代理、转发和NAT（网络地址转换）。



外部通信

指定集群内服务对外暴露的端口，集群内的所有节点都监听该端口，请求到任意节点的**Node IP: nodePort**均可访问到集群内对应服务。

容器化微服务架构 = Docker + Kubernetes

docker

- 进程级内核 轻量级虚拟化技术
- 跨平台，与具体底层系统架构解耦合
- Namespace+Cgroup 计算资源隔离机制
- 细粒度：额外开销

kubernetes

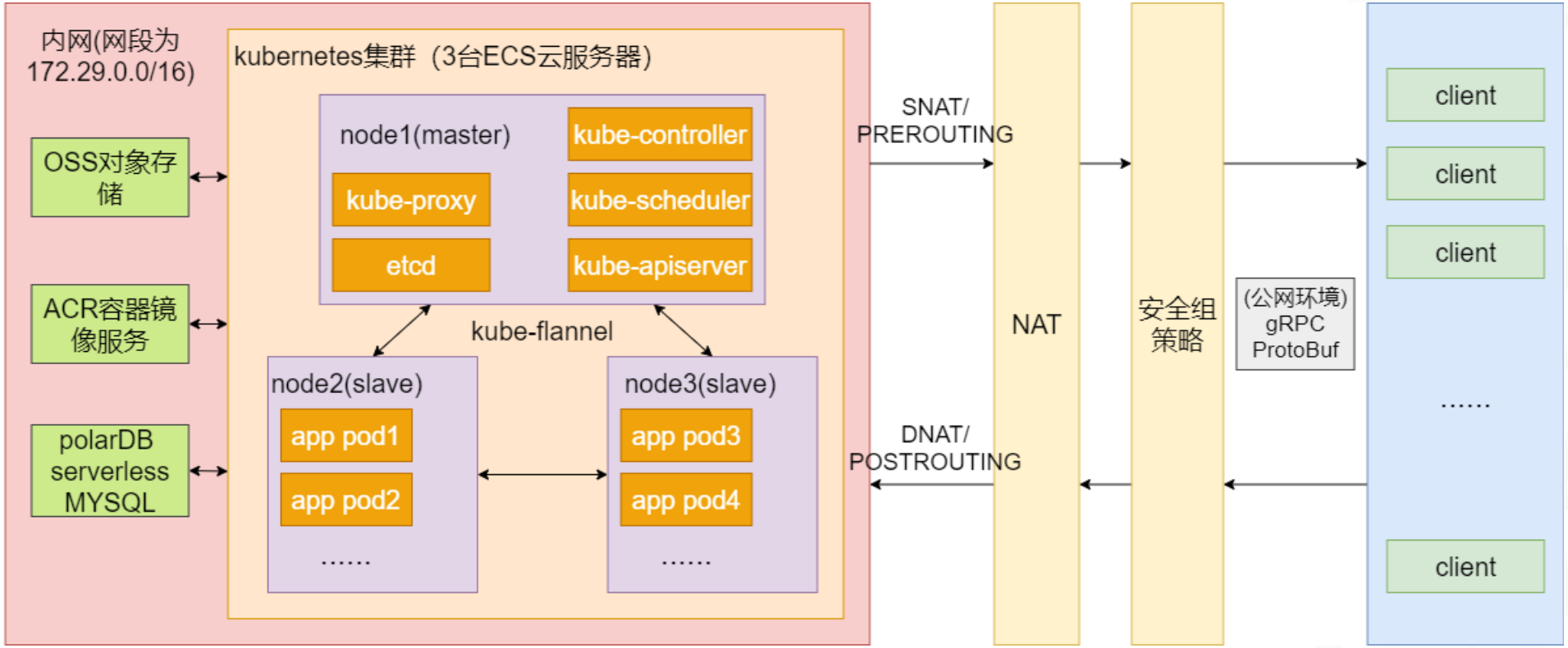
- 去中心化 分布式部署
- 高内聚 低耦合
- 容器集群
- 自动部署 节点监控 智能调度 负载均衡

10CPU*1 > 1CPU*10

分布式集群的**根本目标**：使得服务性能随着计算资源而**线性**增长。

云计算的基础就是建立在**虚拟化的容器集群**之上的，由管控统一进行资源的分配调度。

阿里云计算平台上的Kubernetes集群部署形态



安全组策略：流量清洗与过滤。

NAT（Network Address Translation，网络地址转换）：基于iptables，**DNAT**（目标地址转换）和**SANT**（源地址转换），实现内网和公网之间的通信。

在阿里云计算平台上搭建Kubernetes集群

ECS云服务器

节点主机名	内网IP	公网IP	身份
Node1	172.29.1.1/16	不固定(随机分配)	Master
Node2	172.29.1.2/16	不固定(随机分配)	Slave1
Node3	172.29.1.3/16	不固定(随机分配)	Slave2

```
root@node3:/etc/docker# kubeadm config images list
I0424 03:16:32.389705   10929 version.go:255] remote version is much newer: v1.27.1; falling back to: stable-1.23
k8s.gcr.io/kube-apiserver:v1.23.17
k8s.gcr.io/kube-controller-manager:v1.23.17
k8s.gcr.io/kube-scheduler:v1.23.17
k8s.gcr.io/kube-proxy:v1.23.17
k8s.gcr.io/pause:3.6
k8s.gcr.io/etcd:3.5.1-0
k8s.gcr.io/coredns/coredns:v1.8.6
```

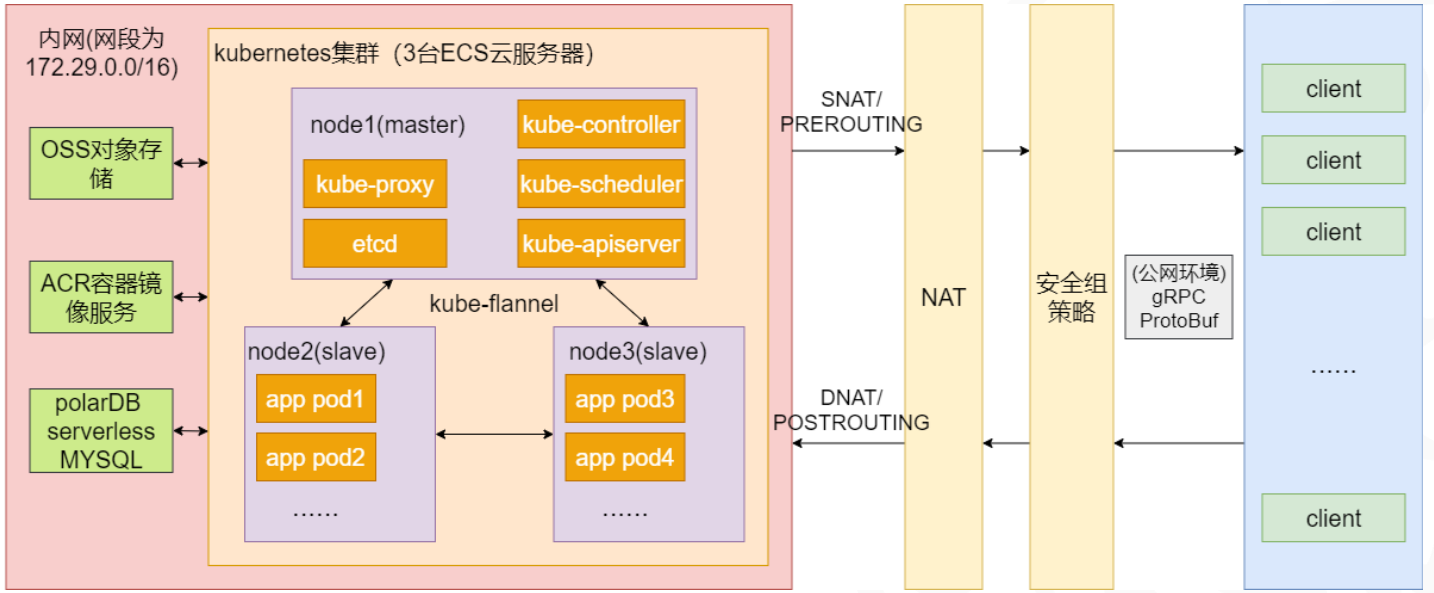
```
root@node1:~# kubectl get nodes
NAME    STATUS   ROLES                  AGE   VERSION
node1   Ready    control-plane,master   4d8h  v1.23.6
node2   Ready    <none>                 4d8h  v1.23.6
node3   Ready    <none>                 94s   v1.23.6
```

1.禁用ufw防火墙和swap交换分区。
2.基于以上网络拓扑架构，安装并配置Docker和Kubernetes开源软件。
3.从**公有仓库**中拉取所需的镜像。
4.使用Flannel**网络插件**使得集群节点之间可以互相通信。

在阿里云计算平台上搭建附属服务

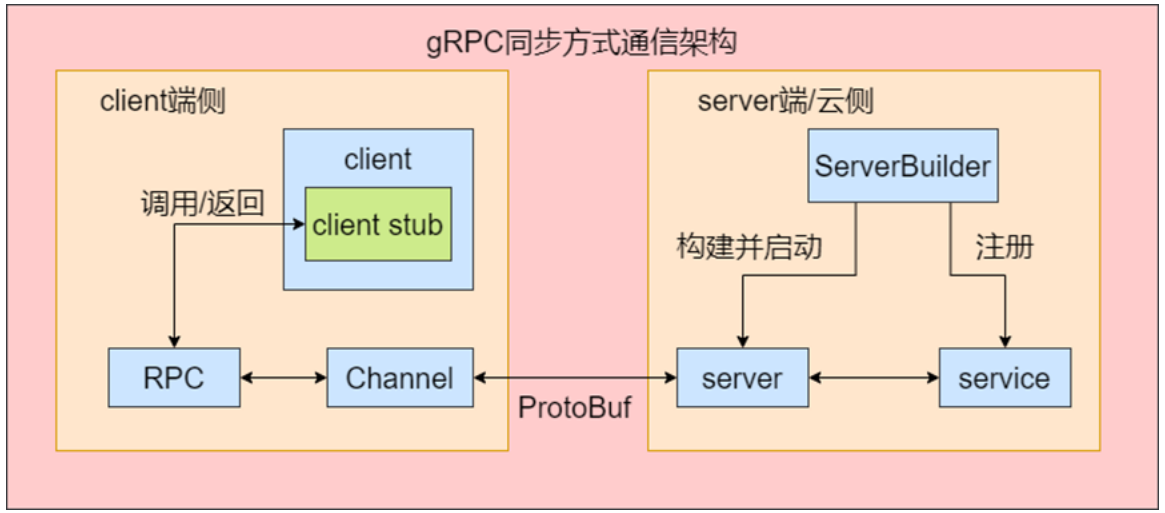
附加服务

- OSS对象存储
- ACR 容器镜像服务
- 云原生数据库 PolarDB Serverless MySQL



各项云平台服务均部署在**内网**环境下，可以直接互相连通。具有高效率，低时延的优势，服务之间相互调用**不需要**经由内网-公网-内网的转换。

gRPC 高性能分布式框架




```cpp
void RunServer()
{
    std::string server_address(USER_ORDER_SERVER);
    MyUserOrderServiceImpl service;
    service.MysqlStart();

    ServerBuilder builder;
    // Listen on the given address without any authentication mechanism.
    builder.AddListeningPort(server_address, grpc::InsecureServerCredentials());
    // Register "service" as the instance through which we'll communicate with
    // clients. In this case it corresponds to an *synchronous* service.
    builder.RegisterService(&service);
    // Finally assemble the server.
    std::unique_ptr<Server> server(builder.BuildAndStart());
    LOG(INFO) << "User Order Server listening on " << server_address;

    // Wait for the server to shutdown. Note that some other thread must be
    // responsible for shutting down the server for this call to ever return.
    server->Wait();
}
```

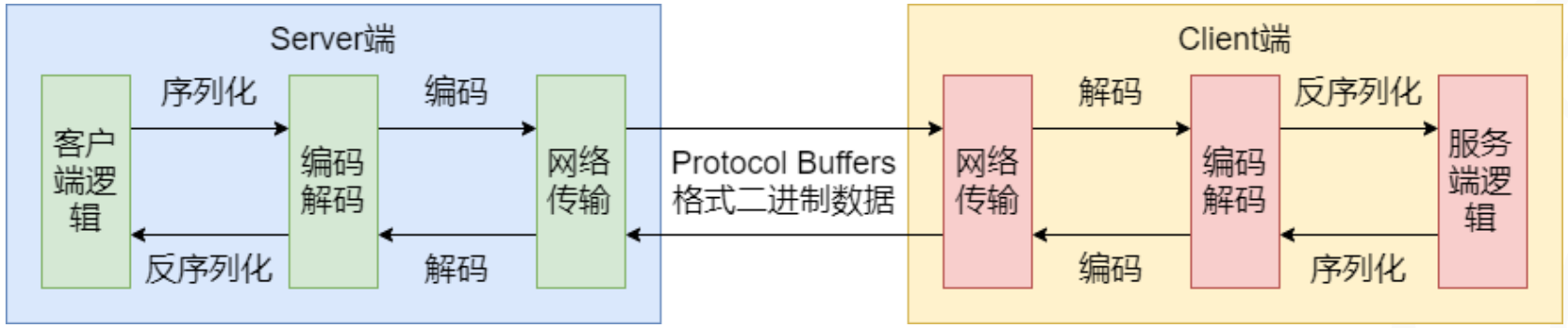
```cpp
    if (x > (double)(GPR_ATM_MAX)) return GPR_ATM_MAX;
    return static_cast<gpr_atm>(x);
  }

static gpr_atm timespec_to_atm_round_up(gpr_timespec ts) {
    ts = gpr_time_sub(ts, g_start_time);
    double x = GPR_MS_PER_SEC * static_cast<double>(ts.tv_sec) +
               static_cast<double>(ts.tv_nsec) / GPR_NS_PER_MS +
               static_cast<double>(GPR_NS_PER_SEC - 1) /
                   static_cast<double>(GPR_NS_PER_SEC);
    if (x < 0) return 0;
    if (x > (double)(GPR_ATM_MAX)) return GPR_ATM_MAX;
    return static_cast<gpr_atm>(x);
```

RPC（Remote Procedure Call），中文译名为**远程过程调用**，是分布式系统常用的通信技术。

gRPC是由Google公司开发的开源、**跨语言**、**跨平台**的RPC框架，基于HTTP/2网络通信协议和Protocol Buffers序列化/反序列化协议进行开发。

Protocol Buffers 序列化/反序列化协议



技术特点

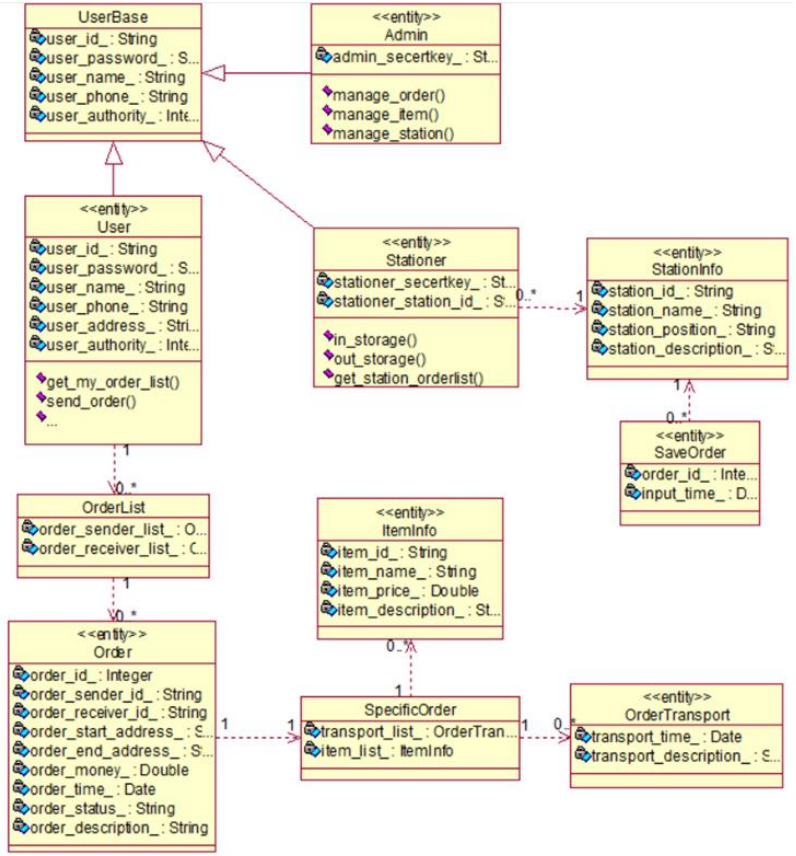
- 二进制序列化编码
- 安全性：对称加密
- 跨语言、跨平台

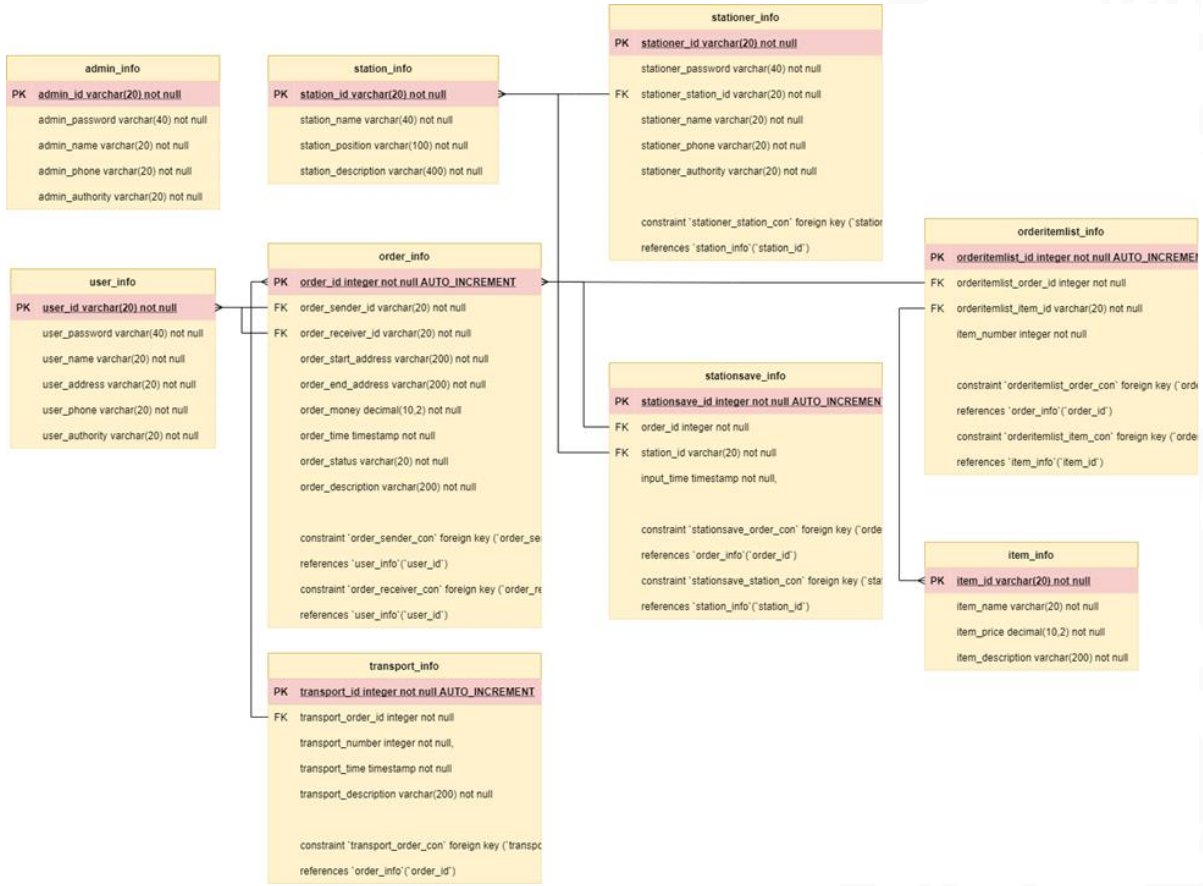
只需要编写**.proto协议文件**，使用代码生成技术产出对应版本的C++文件和gRPC**接口**

```
service UserOrder {
//get message
    rpc GetOrderList(UserId) returns (OrderList); //包含detail
    rpc GetSpecificOrder(OrderId) returns (SpecificOrder);

//post message
    rpc CreateOrder(OrderDetail) returns (BaseReply); //寄件
    rpc UpdateOrderState(UpdateOrderStateRequest) returns (BaseReply); //确认收件
}
```

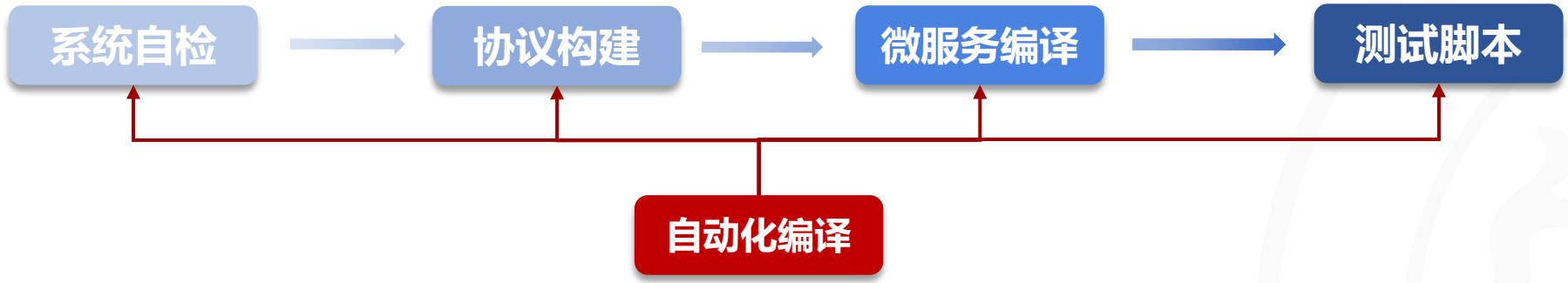
软件工程&系统架构设计





需求分析 → 数据库设计 → 类图设计 → 系统架构 →

Server端微服务编译架构



虚拟开发及编译环境

VMware workstation 15.5 pro + Linux Ubuntu 20.04.5 LTS

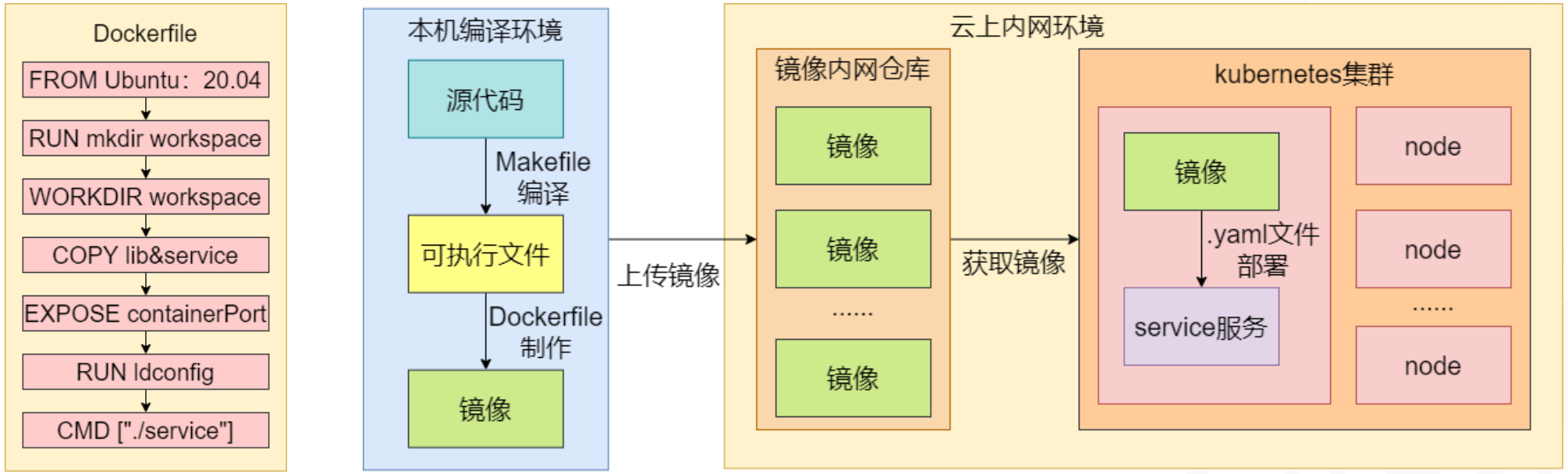
云计算平台运行环境

ECS云服务器： Linux Ubuntu 20.04.5 * 3 (1 Master + 2 Slave)

本地环境

Windows 10 21H2 64bit

镜像制作与集群微服务部署



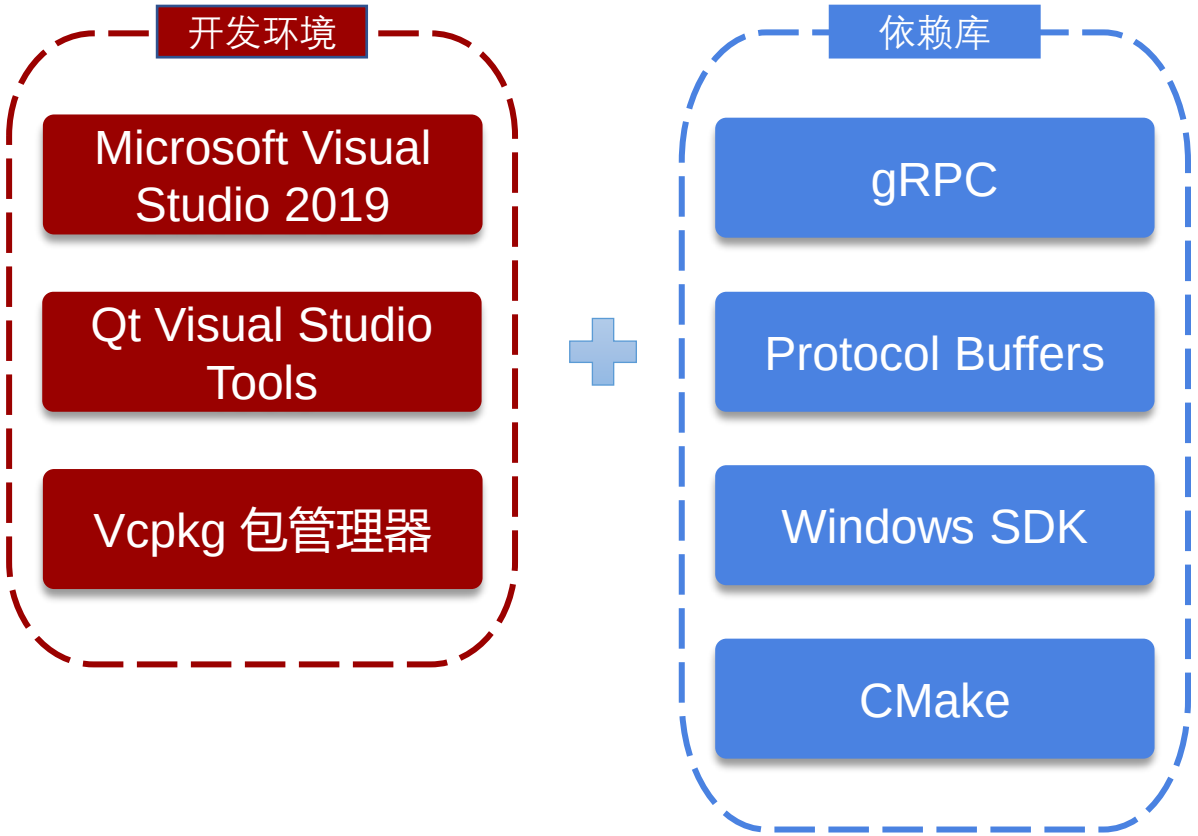
镜像制作：Dockerfile

Dockerfile是一种镜像制作专用的脚本文件，关于镜像的构建过程，是一种**千层饼式**的构建，每执行一次命令都是一次镜像的堆叠，在原有的镜像上添加一层。

集群微服务部署：yaml配置文件

Deployment部分在集群中部署一组Pod（容器组），而Service完成对一组Pod的统一管理。在Kubernetes集群中使用kubectl工具进行部署。

Client端实验环境



Vcpkg包管理器

一站式的环境管理和配置，自动管理依赖版本，解决以下常见的开源软件冲突：

- 不同版本号
- debug/release
- x86/x64/…

系统运行界面

系统功能

- 订单管理
- 货物管理
- 物流管理
- 站点管理
- ……

本质：C-S架构

用户

- 普通用户
- 工作人员
- 站点人员

Client → ProtoBuf → Server → Database

完备的日志系统——Google Logging日志系统

INFO (普通)

WARNING (警告)

ERROR (错误)

FATAL (致命)

```
Log file created at: 2023/04/12 11:00:46
Running on machine: ubuntu
Running duration (h:mm:ss): 0:00:00
Log line format: [IWEF]yyyymmdd hh:mm:ss.uuuuuu threadid file:line] msg
I20230412 11:00:46.108232 13955 mysql_common.cc:28] mysql_init successed
I20230412 11:00:46.111723 13955 mysql_common.cc:39] mysql_real_connect to localhost, database Logistics_System, user root successed
I20230412 11:00:46.117744 13955 admin_server.cc:255] Admin Server listening on 0.0.0.0:50055
I20230412 11:00:48.098058 13958 mysql_common.cc:169] mysql_query success
I20230412 11:00:48.098124 13958 admin_server.cc:66] GetAllItemList query succeed, admin_id_:13544444444
I20230412 11:00:50.963546 13986 mysql_common.cc:169] mysql_query success
I20230412 11:00:50.963593 13986 admin_server.cc:96] GetAllStationList query succeed, admin_id_:13544444444
I20230412 11:00:53.307695 13958 mysql_common.cc:169] mysql_query success
```

```
Log file created at: 2023/04/04 12:41:35
Running on machine: ubuntu
Running duration (h:mm:ss): 0:00:00
Log line format: [IWEF]yyyymmdd hh:mm:ss.uuuuuu threadid file:line] msg
E20230404 12:41:35.928225 111715 mysql_common.cc:98] mysql_insert failed(You have an error in your SQL syntax; check the manual
E20230404 12:41:35.929080 111715 mysql_common.cc:99] inst:INSERT INTO orderitemlist_info VALUES (, 13, '1', 2);
```

Google公司开发的Google Logging（GLOG）日志库，基于C++14标准。

细粒度：具体到具体到源代码文件的某一行，便于对记录信息进行精准定位。

分级记录：按照严重程度进行划分。

单元测试——google/benchmark

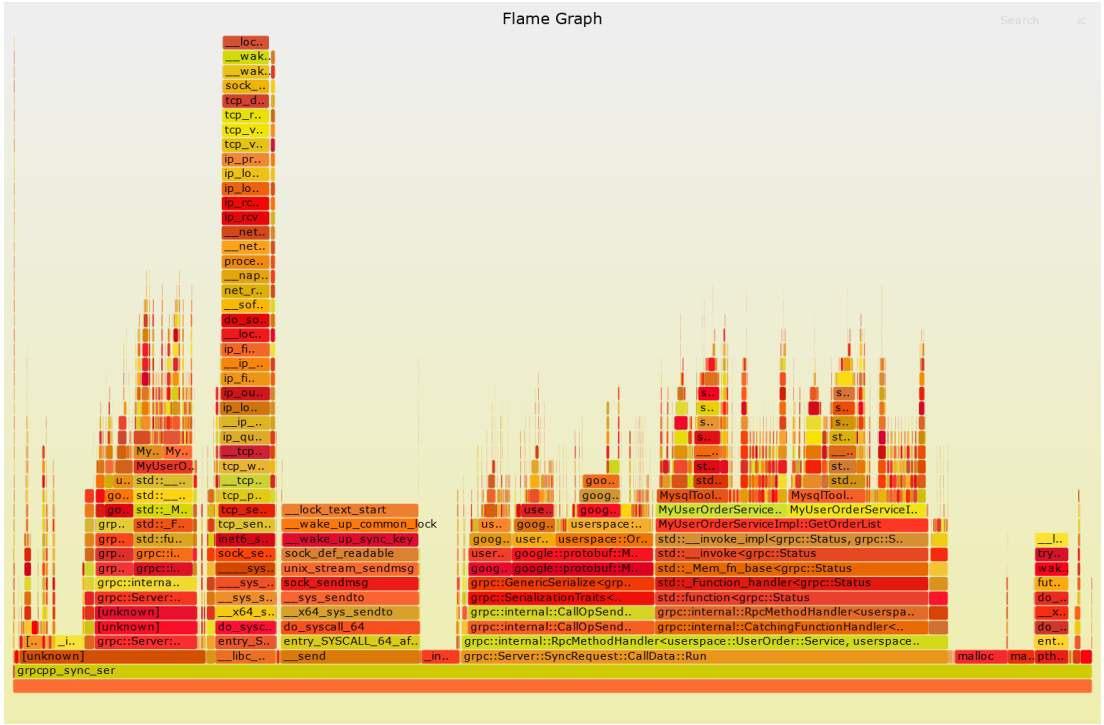
测试机CPU参数（主频、三级缓存、负载）：

Run on (2 X 2592 MHz CPU s)
CPU Caches:
  L1 Data 32 KiB (x2)
  L1 Instruction 32 KiB (x2)
  L2 Unified 256 KiB (x2)
  L3 Unified 12288 KiB (x1)
Load Average: 0.73, 0.80, 0.89

以上万数据量级为单位的高并发时间是线性增长的!

测试名	测试数量级	总时间/ns	CPU时间/ns	测试轮数	平均CPU时间/ns
GetOrderList	10	7676175	1147713	644	114771.30
GetOrderList	100	73207912	11116059	63	111160.59
GetOrderList	1000	749699961	112163170	6	112163.17
GetOrderList	10000	7561785564	1135731425	1	113573.14

系统级性能测试——火焰图Flame Graph



尖峰：程序在当前的函数中时间占比较少，说明该处函数组**不是**影响程序性能的关键因素。

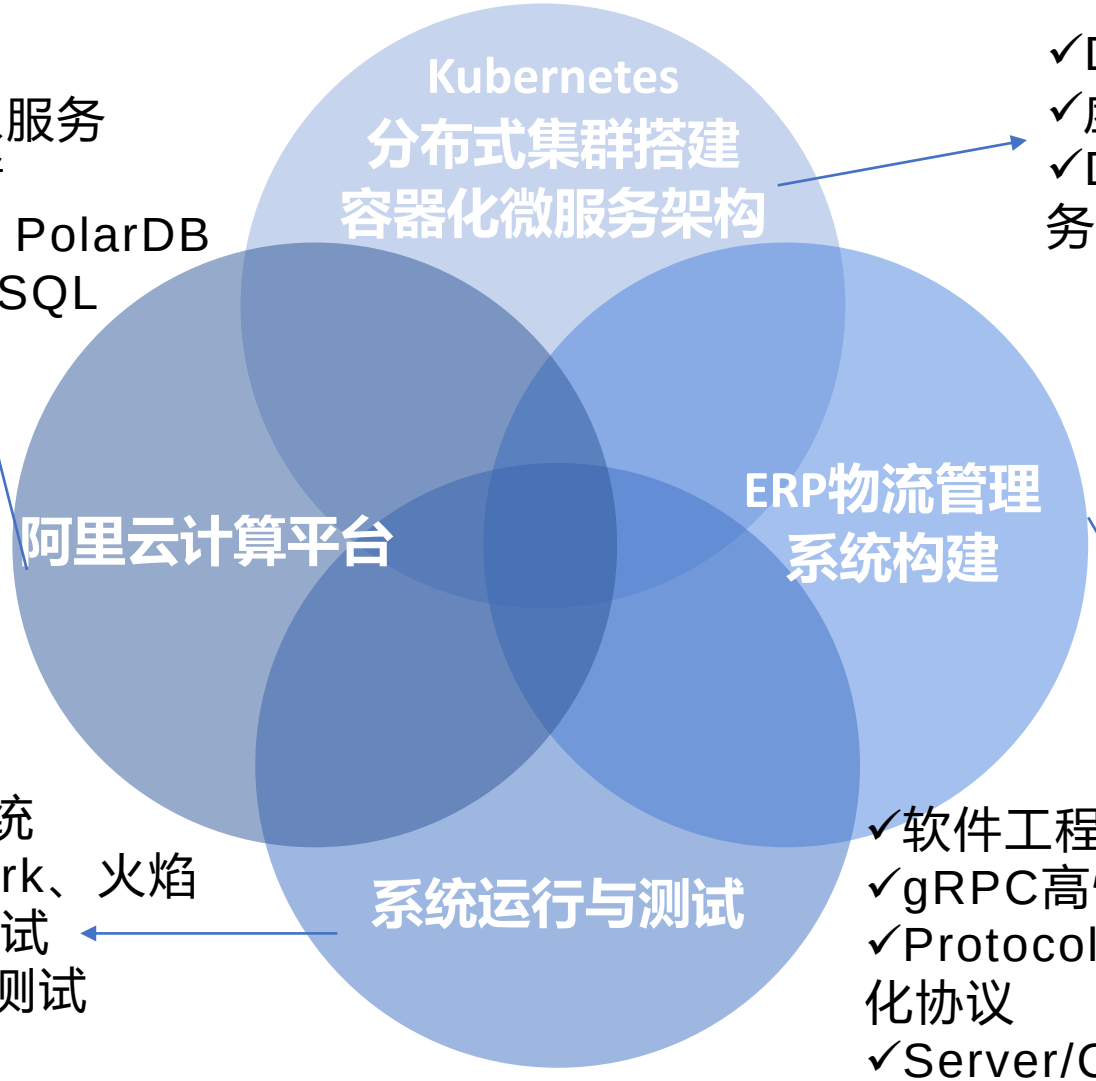
平顶：程序在当前的函数中运行时间占比较大，说明该处存在着占用时间的**性能瓶颈**或需要耗费大量CPU运行时间的重要服务。

统计学性能优化策略

perf命令及原理

使用Linux 系统原生提供的性能分析工具perf命令，可以在一段时间内对CPU上的嵌套堆栈进行一定频率的**采样**，返回CPU正在执行的函数名以及调用栈，监控程序的**堆栈嵌套调用**以及每层堆栈函数的**CPU运行时间**。基于以上perf的统计结果，对数据进行进一步分析，生成SVG格式的火焰图。

工作总结



✓ECS云服务器
✓ACR容器镜像服务
✓OSS对象存储
✓云原生数据库 PolarDB Serverless MySQL

✓Docker+Kubernetes
✓虚拟化容器集群搭建
✓Dockerfile镜像制作&微服务编排、封装、部署和发布

✓GLOG日志系统
✓使用benchmark、火焰图等进行量化测试
✓系统高可用等测试

✓软件工程&系统架构设计
✓gRPC高性能分布式框架
✓Protocol Buffers 序列化/反序列化协议
✓Server/Client端代码开发与编译

展望

- 修改集群默认的节点调度算法，通过部署插件的方式**自定义集群调度算法**并部署在集群中，在高负载情况下通过改进调度算法提高集群的服务性能。
- 将gRPC同步通信修改为**异步**方式，以应对更高数据量级的并发场景，达到使用异步通信对流量进行限流削峰的作用。
- 将系统接入**官方**个人信息接口和支付系统**接口**等，从而提高系统的安全性和可靠性，并进一步完善系统的各项功能，便于投入实际使用和提高用户体验。

同濟大學
TONGJI UNIVERSITY

恳请各位老师批评指正

Thanks for Listening

答辩人: 肖鹏飞　　学号: 1953072　　方向: 工业软件与云计算　　指导老师: 曾国荪