



Serverless的介绍及应用

林许亚伦 支晨曦

2019.6.10



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

1

背景

2

Serverless简介

3

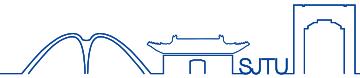
实验结果

4

常见应用场景

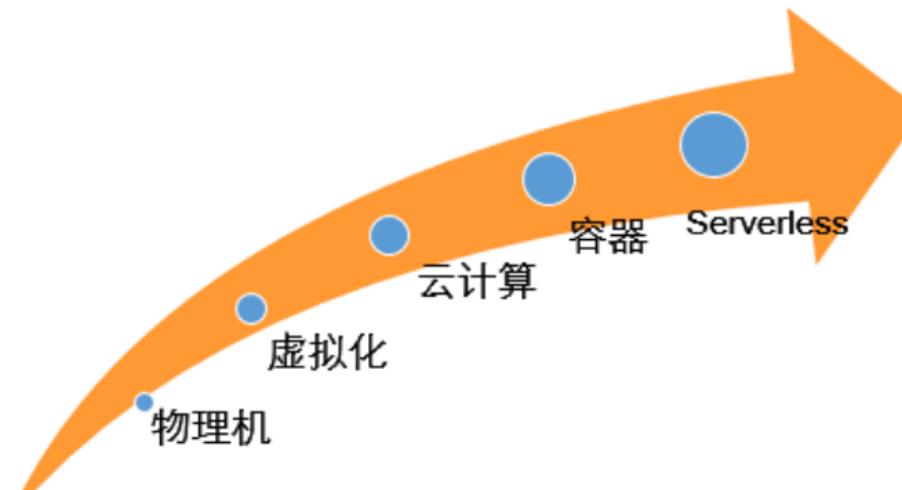


背景 - 系统架构发展历程



- 里程碑事件：

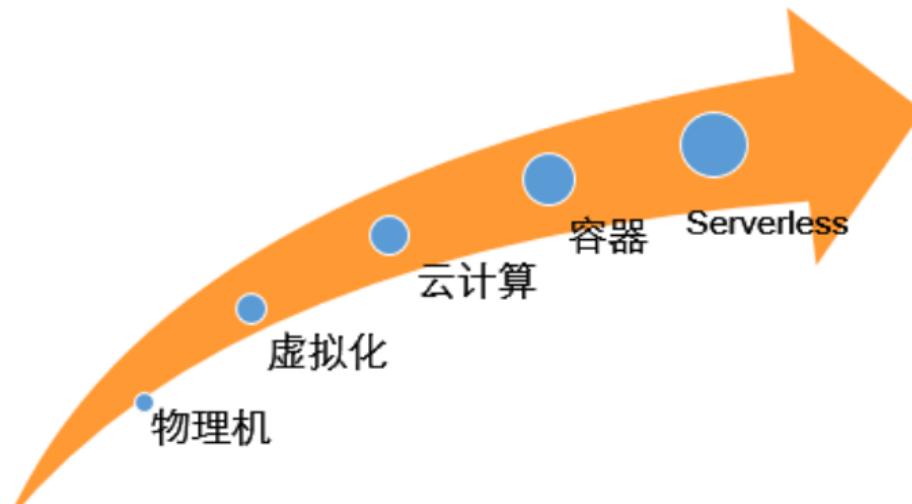
- 通过虚拟化技术将大型物理机虚拟成单个VM资源
- 把虚拟化集群搬到云平台上，只做简单运维
- 把每一个VM按照运行空间最小化原则切分成更细的Docker容器
- 基于Docker容器构建不需要任何运行环境，仅需要编写核心代码的Serverless架构



背景 - 系统架构发展历程



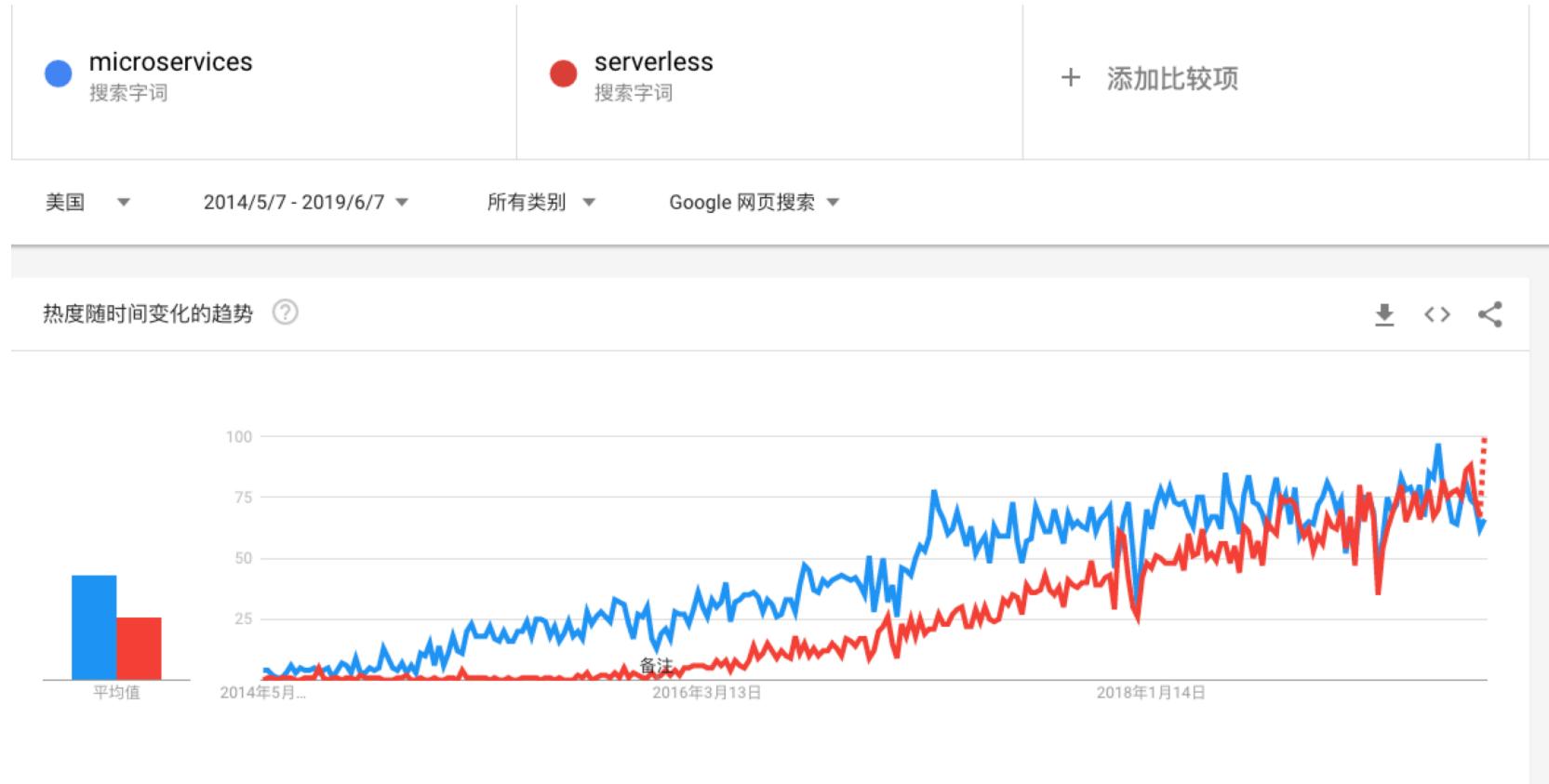
- 系统演进的特点：
 - 硬件资源使用颗粒度变小
 - 资源利用率越来越高
 - 运维工作逐步减少
 - 业务更聚焦在代码层面



背景 – Serverless的发展



- 微服务在2014年开始流行起来
- Serverless在2016年起，收到开发者的关注



1

背景

2

Serverless简介

3

实验结果

4

常见应用场景





Serverless简介

- Serverless是指开发者在构建和运行应用时无需管理服务器等基础设施
- 应用被解耦为细粒度的函数，函数作为部署和运行的基本单位。
- 代码由事件触发，用户只为实际使用的资源付费
- 常见的Serverless服务提供厂家
 - Amazon AWS Lambda
 - Google Cloud Functions
 - Microsoft Azure Functions
 - IBM OpenWhisk
 - ...



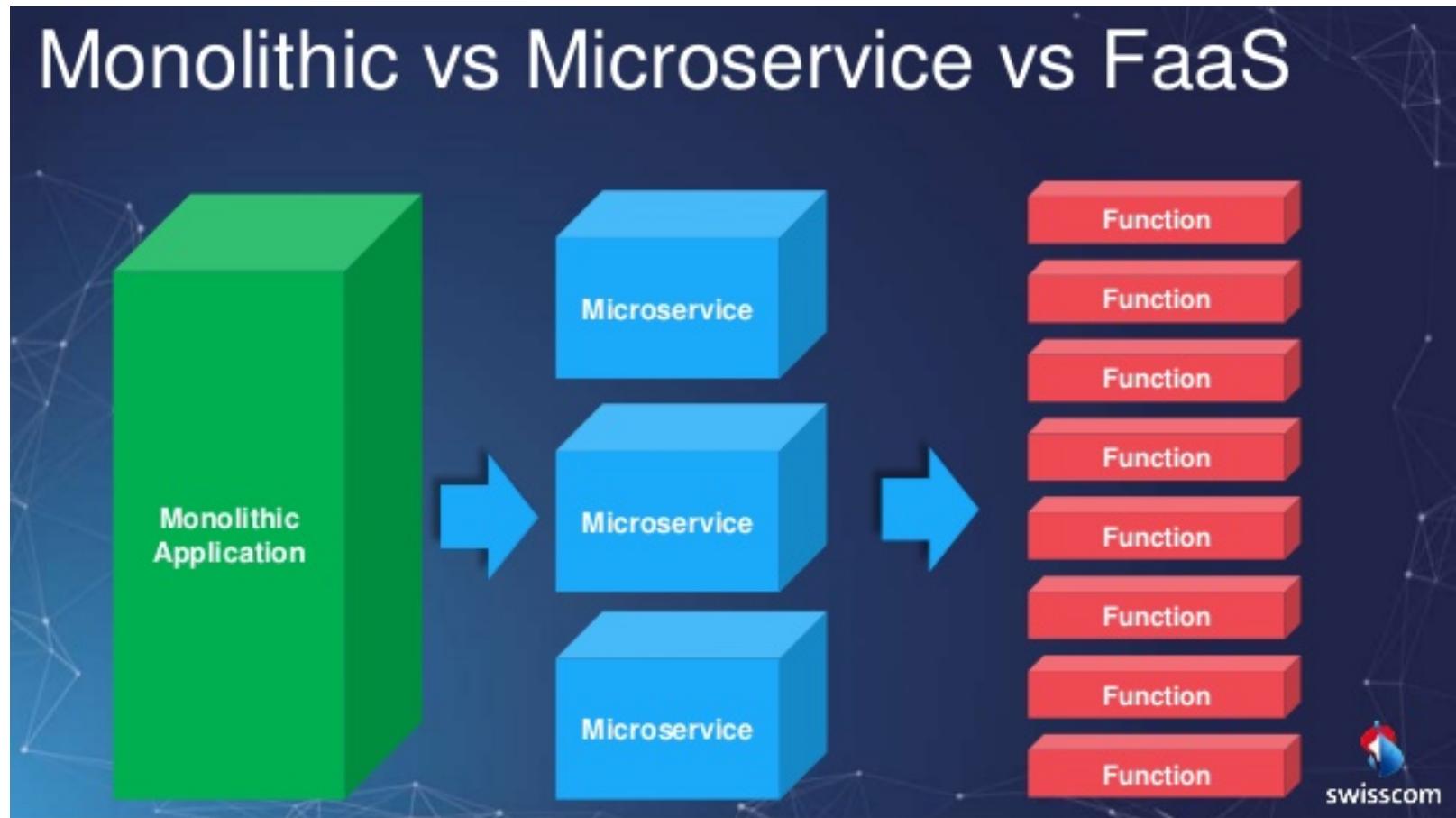


Serverless简介 – 特点

- 在Serverless应用中，开发者只需要专注于业务，而不用担心运维问题
- Serverless是真正的按需使用，只有请求到来时才运行
- Serverless按照运行时间和内存计价
- Serverless严重依赖于特定的云平台和第三方服务



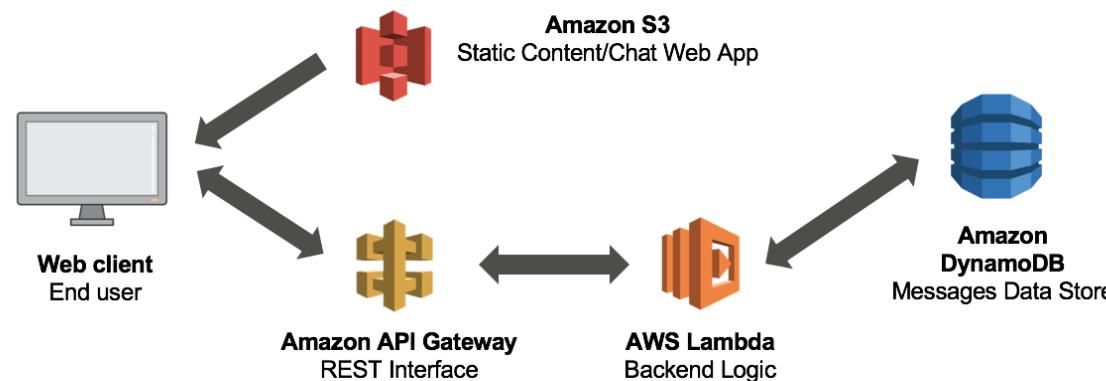
Serverless简介 – 系统划分粒度





Serverless简介 – 应用的组成

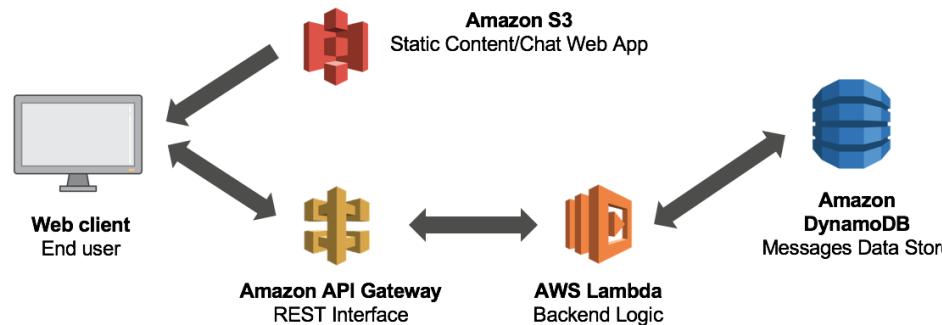
- 一个基于AWS的Serverless应用里，应用的组成是：
 - 网关API Gateway
 - 计算服务Lambda
 - 基础设施管理CloudFormation
 - 静态存储S3
 - 数据库DynamoDB
 -





Serverless简介 – 事件驱动

- Serverless中的服务会在指定事件发生后再被执行
- 这些服务只有在被执行时才被计费
- 以创建博客为例
 - 请求来到API Gateway, API Gateway计费器+1
 - 请求来到Lambda, 进行数据处理, 如生成ID、创建时间等等, Lambda计费器+1
 - Lambda计算完后, 把数据存储到DynamoDB上, DynamoDB计费器+1
 - 生成的静态博客存储到S3, S3只在使用时按存储收费



Serverless简介 – 优缺点总结



- Serverless的优点：
 - 降低启动成本
 - 减少运营成本
 - 降低开发成本
 - 实现快速上线
 - 更快的部署流水线
 - 更快的开发速度
 - 系统安全性更高
 - 适应微服务架构
 - 自动扩展能力
 -
- Serverless的缺点：
 - 不适合长时间运行应用
 - 完全依赖第三方服务
 - 冷启动时间
 - 缺乏调试和开发工具
 - 构建复杂
 - 语言版本落后
 -



1

背景

2

Serverless简介

3

实验结果

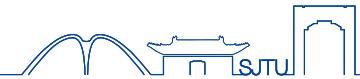
4

常见应用场景





Serverless 实验配置



Lambda 定价详情

区域: 亚太地区（新加坡）◆

Lambda 每次针对事件通知或调用（包括控制台中的测试调用）开始执行代码，就计算一个请求。您将根据所有函数的请求总数为每个请求支付 0.0000002 USD。

持续时间从您的代码开始执行时算起，到其返回或终止为止，舍入到最近的 100 毫秒。价格取决于您为函数分配的内存量。在 AWS Lambda 资源模型中，您可以选择用于函数的内存量，这会分配等比例的 CPU 计算能力和其他资源。内存大小的增加会导致函数可用的 CPU 的等效增加。要了解更多信息，请参阅我们的[功能配置文档](#)。

Lambda 免费套餐包含每月 100 万个免费请求以及每月 400000GB-秒的计算时间。

免费套餐

100 万个请求

每月

400000GB-秒

计算时间（每月）。

Lambda 免费套餐在 AWS 免费套餐 12 个月的期限到期后不会自动过期，而是无期限地提供给现有的和新的 AWS 客户。

请求

100 万个免费请求

每月前 100 万个请求免费。

此后每 100 万个请求 0.20 USD

每个请求 0.0000002 USD。

持续时间

每月 400000GB-秒的免费时间

每月前 400000GB-秒且最多 320 万秒的计算时间免费。

此后使用的每 GB-秒 0.0000166667 USD

价格取决于您为函数分配的内存量。



Serverless 实验配置

▼ Designer

添加触发器

选择以下列表中的某个触发器以将其添加到您的函数。

API Gateway

AWS IoT

Application Load Balancer

CloudWatch Events

CloudWatch Logs

CodeCommit

从左侧的列表中添加触发器

helloWorld

Layers (0)

API Gateway 已保存

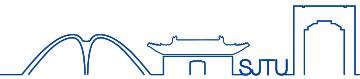
S3 已保存

Amazon CloudWatch Logs

此处将显示该函数角色对其具有访问权限的资源



Serverless 实验配置



函数代码 信息

代码输入种类 在线编辑 运行语言 Node.js 10.x 处理程序 信息 index.handler

File Edit Find View Go Tools Window Environment

index.js

```
1 exports.handler = async (event) => {
2     let dynamicHTML = '<p>Hey Unknown!</p>';
3     // check for GET params and use if available
4     if (event.queryStringParameters && event.queryStringParameters.name) {
5         dynamicHTML = `<p>Hey ${event.queryStringParameters.name}!</p>`;
6     }
7
8     const html = `
9         <html>
10             <style>
11                 h1 { color: #73757d; }
12             </style>
13             <body>
14                 <h1>Hello AWS Lambda!</h1>
15                 ${dynamicHTML}
16             </body>
17         </html>`;
18
19     const response = {
20         statusCode: 200,
21         headers: {
22             'Content-Type': 'text/html',
23         },
24         body: html,
25     };
26
27     return response;
28 };
29 
```

29:1 JavaScript Spaces: 4



Serverless 实验配置



环境变量

您可使用键/值对的形式定义可从函数代码访问的环境变量。这对于存储配置设置而无需更改函数代码非常有用。[了解更多信息](#)

键	值	删除
---	---	----

▶ 加密配置

标签

您可以使用标签对功能进行分组和筛选。标签包含一个区分大小写的键值对。[了解更多信息](#)

键	值	删除
---	---	----

执行角色

选择定义您的函数权限的角色。要创建自定义角色, 请转至 [IAM 控制台](#)。

使用现有角色	▼
--------	---

现有角色

选择您创建的现有角色以与此 Lambda 函数结合使用。此角色必须有权将日志上传到 Amazon CloudWatch Logs。

service-role/helloWorld-role-2t3mccwu	▼	
---------------------------------------	---	--

在 IAM 控制台上 [查看 helloWorld-role-2t3mccwu 角色](#)。

基本设置

描述

--

内存 (MB) 信息

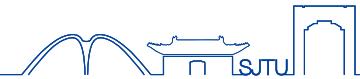
您函数被分配的 CPU 是与配置的内存成比例的。

128 MB

超时 信息

0	分钟	3	秒
---	----	---	---

Serverless 实验配置



helloWorld

限制 版本控制 操作 test 测试 保存

执行结果: 成功 (日志)

详细信息

以下区域显示函数执行所返回的结果。详细了解有关从您的函数返回结果的信息。

```
{ "statusCode": 200, "headers": { "Content-Type": "text/html" }, "body": "\n      <html>\n          <style>\n              h1 { color: #73757d; }\n          </style>\n          <body>\n              <h1>Landing Page</h1>\n\n              <p>Hey Unknown!</p>\n          </body>\n      </html>" }
```

摘要

代码 SHA-256	请求 ID
YONwJTxOP7fNRDVkGDIWLx4hAyQDczJIExOj8Bw2bM=	847884f9-b4e7-4cf5-a2eb-97143aa936b0
持续时间	计费时间
132.07 ms	200 ms
配置的资源	最大已用内存
128 MB	73 MB

日志输出

以下部分显示代码中的日志记录调用。这些调用与此 Lambda 函数所对应的 CloudWatch 日志组中的一行对应。单击此处 以查看 CloudWatch 日志组。

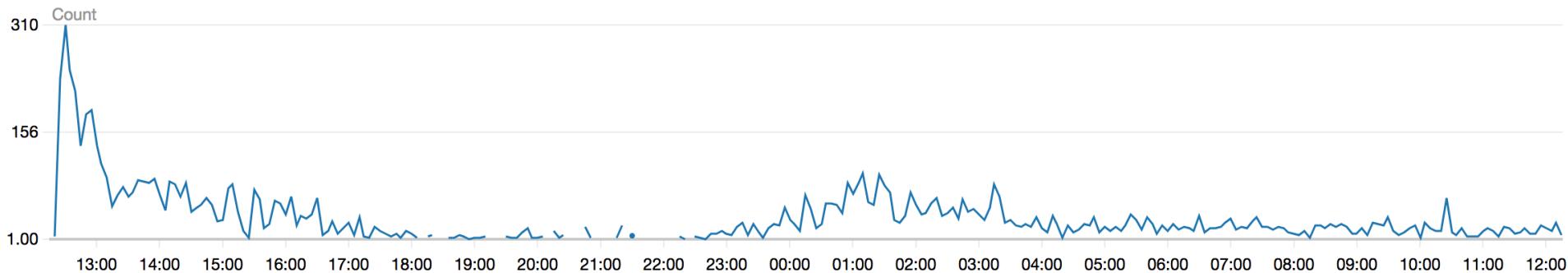
```
START RequestId: 847884f9-b4e7-4cf5-a2eb-97143aa936b0 Version: $LATEST
END RequestId: 847884f9-b4e7-4cf5-a2eb-97143aa936b0
REPORT RequestId: 847884f9-b4e7-4cf5-a2eb-97143aa936b0 Duration: 132.07 ms Billed Duration: 200 ms Memory Size: 128 MB Max Memory Used: 73 MB
```



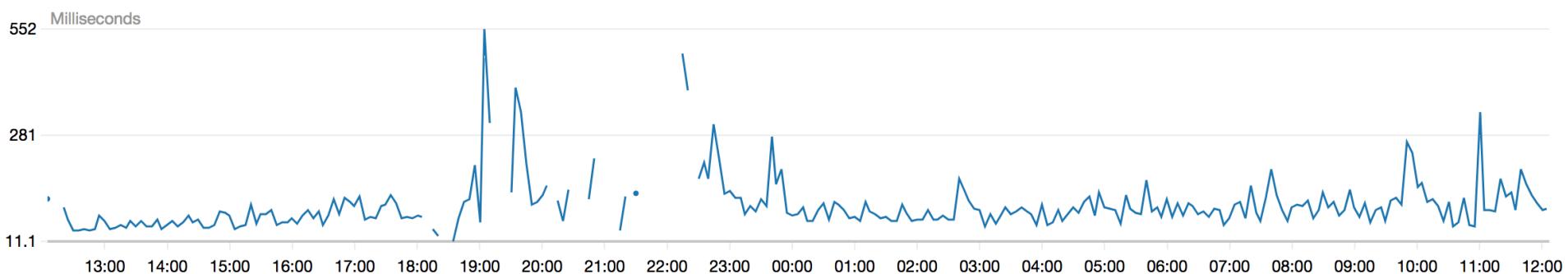
Serverless实验结果



- Serverless请求统计



- Serverless请求时间



1

背景

2

Serverless简介

3

实验结果

4

常见应用场景

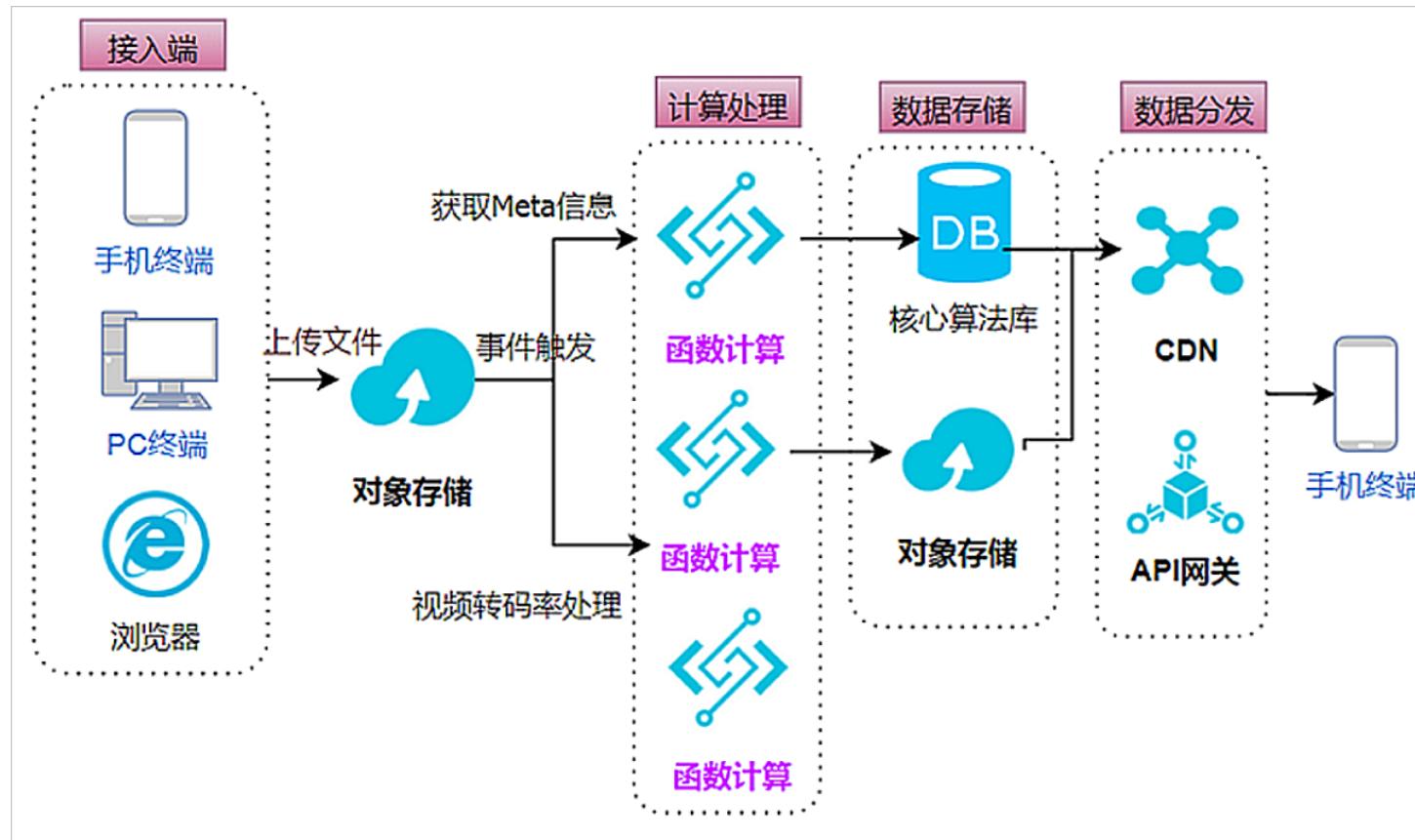




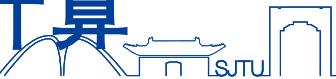
Serverless常见应用场景



- 事件触发计算能力 – 视频处理



Serverless常见应用场景 – 事件触发计算

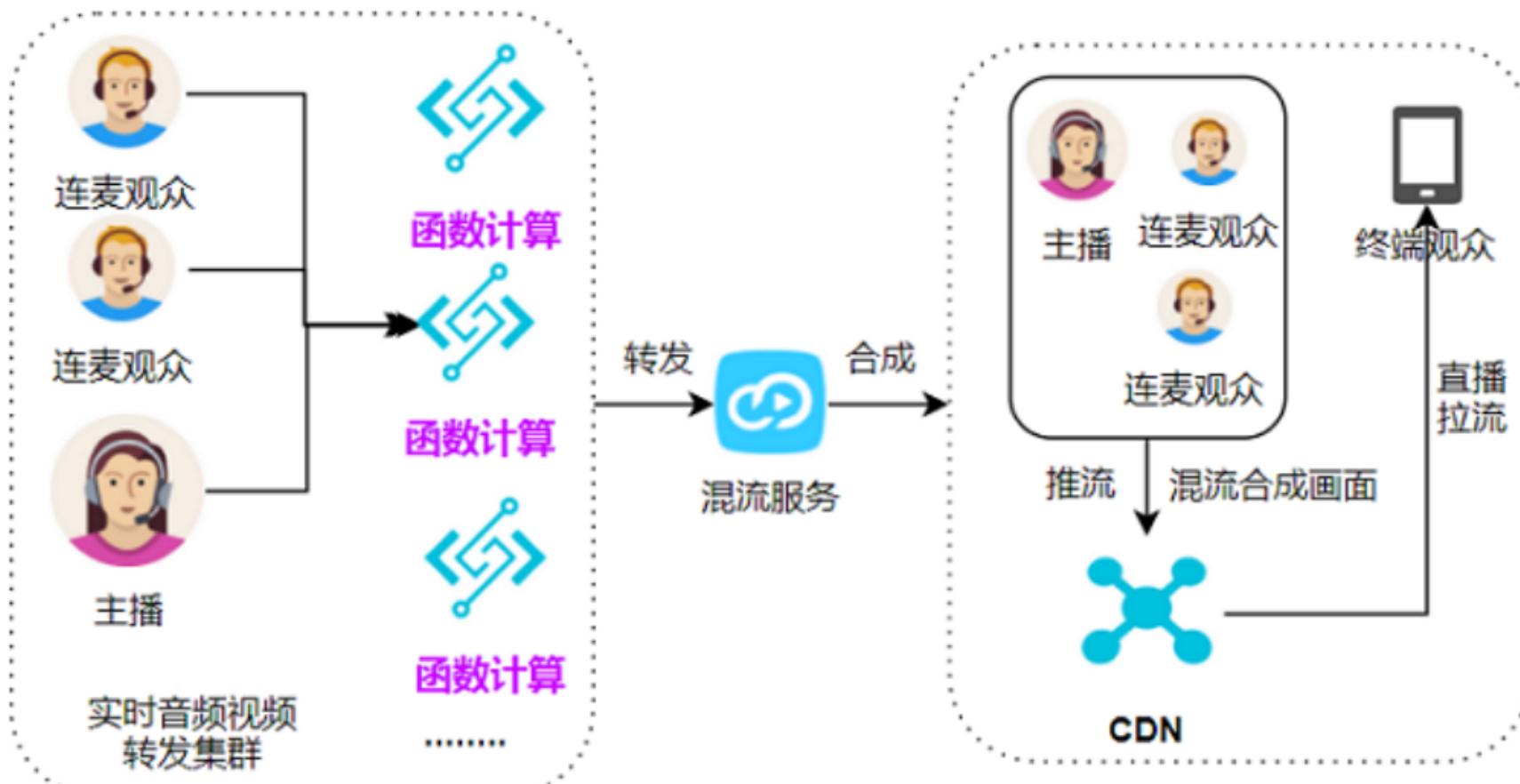


- 常规做法：
 - 设置消息通道接收事件，并编写业务代码
 - 购买服务器资源做后端处理
 - 设计一套多并发框架完成业务上传文件峰值的处理
 - 开通多个产品，并调用SDK代码进行业务交互
- Serverless解法：
 - 在控制台上配置事件源通知，编写业务代码
 - 代码写到函数计算里，不需要管理软件硬件环境
 - 业务高峰期函数计算会动态伸缩，无需管理
 - 内置打通多款产品，简单配置就可以无缝对接

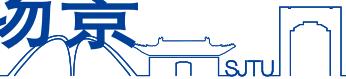


Serverless常见应用场景

- 弹性扩容能力 – 视频多人连麦场景



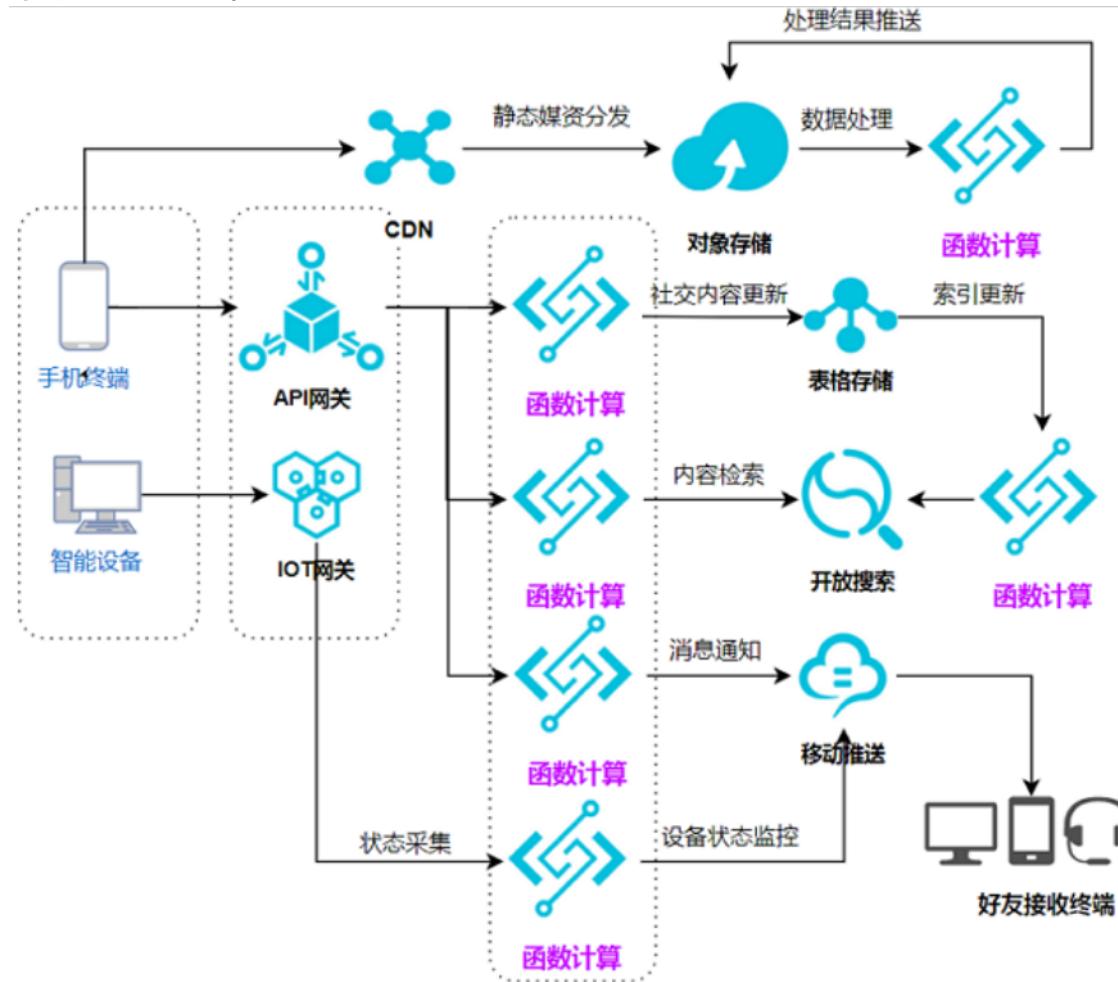
Serverless常见应用场景 – 视频直播场景



- 常规做法：
 - 购买负载均衡应付高并发
 - 购买计算资源做数据处理
 - 业务低谷期需要想办法释放硬件资源来节约成本
 - 多版本维护多套运行环境
- Serverless解法：
 - 把负载分发程序写到函数里
 - 多版本迭代无需更换运行环境，仅仅替代代码版本而已
 - 业务访问需要按需付费，业务低谷期无费用

Serverless常见应用场景

- 物联网数据处理场景



Serverless常见应用场景 – 物联网数据处理

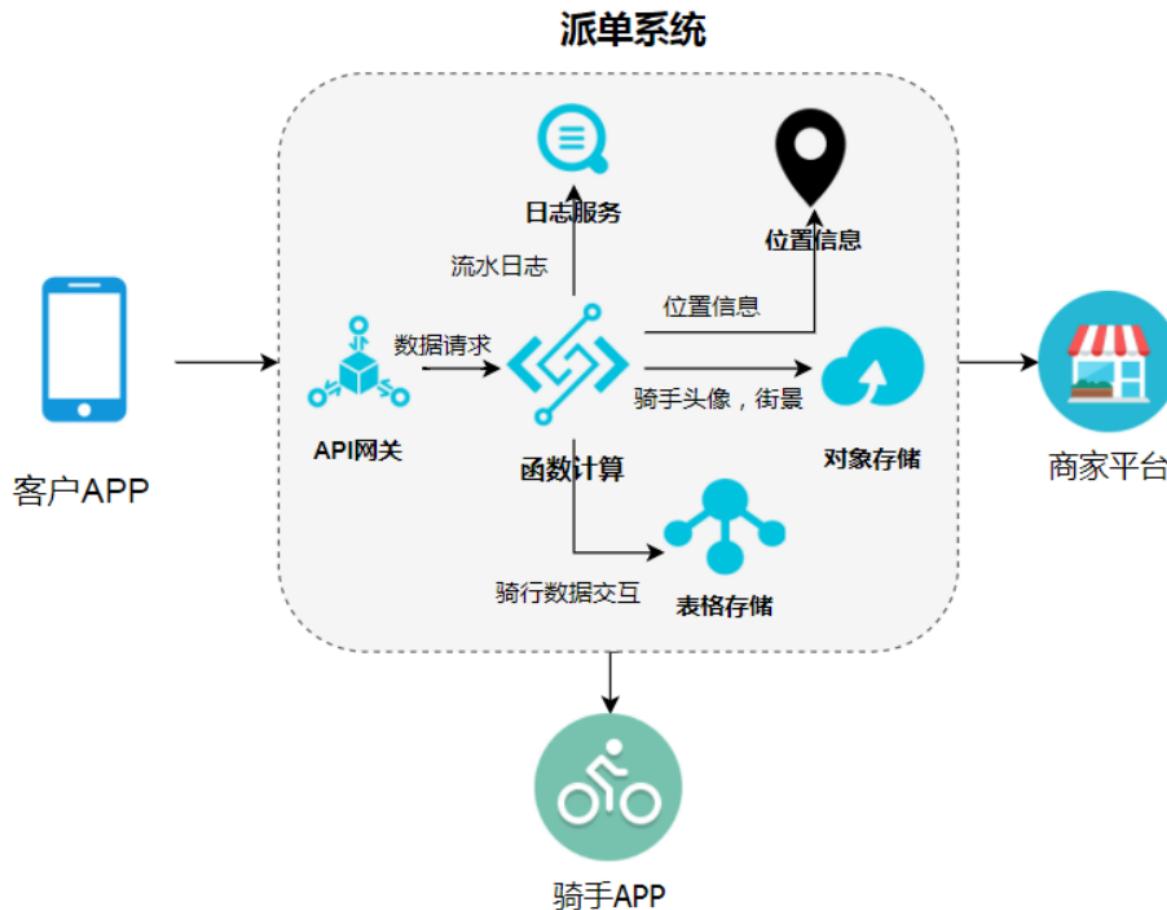


- 常规做法：
 - 设置消息通道接收事件，并编写业务代码
 - 购买服务器资源做后端数据处理
 - 开通多个产品，并调用SDK代码来完成业务交互
 - 维护硬件软件环境
- Serverless解法：
 - 定制IoT平台的事件通知，直接把业务代码写到函数计算中
 - 不需要维护运行环境，用完即可释放
 - 控制台配置，就可以把信息透传给相关产品

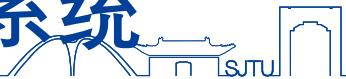


Serverless常见应用场景

- 共享派单系统



Serverless常见引用场景 – 共享派单系统



- 常见做法：
 - 购买多台服务器来支持高峰期的访问，波谷期自行设置释放原则
 - 通过编程方式完成多个产品的交互
 - 为了保证负载均衡，需要购买相关的产品来支撑
 - 维护相关硬件软件环境
- Serverless解法：
 - 定制IoT平台的时间通知，直接把业务代码写到函数计算中
 - 不需要维护运行环境，用完即可释放
 - 控制台配置，就可以把信息透传给相关产品

谢谢！

