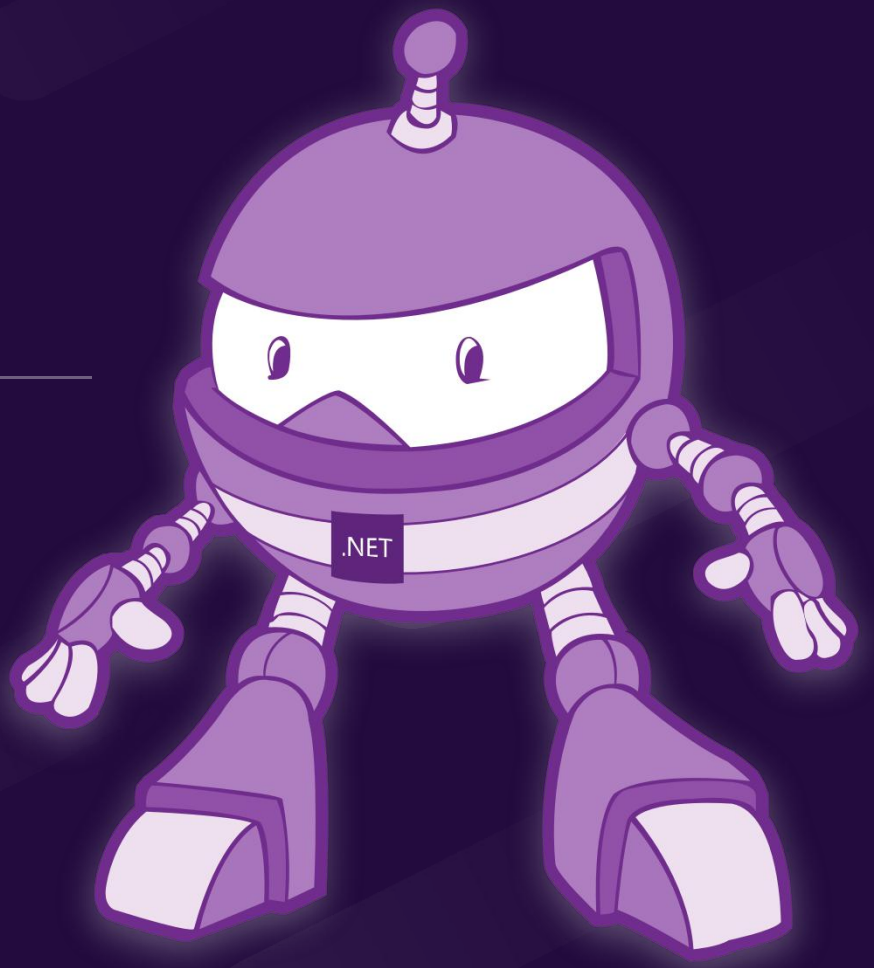
2019中国.NET 开发者峰会
China .Net Conf 2019

.NET Core 2019

.NET大数据实时计算

演讲人：黄国石



数据中通

大数据来源

财报摘要显示，中通快递二季度业务量完成31.07亿件，同比增长46.8%，超出行业平均增速18.5个百分点。市场份额较上年同期提升2.5个百分点达19.9%。

目录
CONTENTS

01 Part

背景介绍

背景介绍

- ✓ 每天一亿多个包裹在路上，产生好几亿数据
- ✓ 末端时效管控，收件时间缩短，提升用户体验
- ✓ 由实时计算平台支撑营运管控系统

- ✓ 每天凌晨，外省大货车进入上海
- ✓ 自动分拣，系统开始跟踪
- ✓ 网点合理安排车辆和人员
- ✓ 派件签收，系统实时反馈（<10min）
- ✓ 管理人员全网监控

背景介绍

- ✓ 全国各地网点，存在地区性差异
- ✓ 系统参数需要不断调整，营运政策改变
- ✓ 计算平台按需重新计算

- ✓ 原始数据主要来自PDA、自动分拣机、狂扫等设备，以及手机APP
- ✓ 公司之间接口对接也会产生大量数据，如订单
- ✓ 还有其它一些中小型IoT物联网设备产生的数据

02 Part

计算平台架构

计算平台架构

系统规模

计算平台以调度系统为核心，40多台虚拟服务器作为计算节点，30多台虚拟服务器作为数据清洗和数据服务，3组数据库（Oracle/MySql），9台Redis（共250实例）。

共支撑70多个系统，700多个计算作业，年计算量万亿级。

技术架构

整个平台设计于2016年，基于2008年某金融公司数据处理思路，采用分片处理数据思想，我们自称蚂蚁算法，实际上就是MapReduce。

系统基于.NET 4.5开发，在2018年初引入.NET Core来做数据清洗、数据服务和数据计算，几乎所有功能模块都同时编译.NET4.5/.NET Core版本，根据部署环境选用，CentOS占80%。

数据流程

各个终端产生的数据汇总后进入公司核心数据库，同时推送RocketMQ消息队列。

频繁使用的核心数据，需要即时消费MQ写入Redis，以获取高吞吐和低延迟。

老系统数据在核心数据库中，可以通过备库按插入时间去抽取，写入Redis。

各计算节点通过专门的RPC网关去Redis分批抽取数据，进行分割计算。

分片实时计算

分片调度

以较小时间片（如5秒）对原始数据进行分片，建立工作任务，多线程分批处理。

运单/订单缓存

全量扫描数据、订单数据，以单号为Key，压缩后写入Redis，有效期30天。

以时间为Key，建立单号列表缓存

数据收集与开放

对接各业务线数据库、消息队列，数据清洗，对外数据开放。

内部数据通信微服务化。

统计与明细

统计数据在内存归并后批量落库；

明细数据经Redis缓冲后批量写入

计算平台

数据规模

Redis共有68亿Key
运单16亿（单号KV）
订单16亿（单号KV）
业务25亿 List/Set
冗余备份11亿
总内存4T，已使用2T

接入应用

配置中心共接入系统模块
188个
活跃应用103个
每日缓存调用量146亿次

系统性能

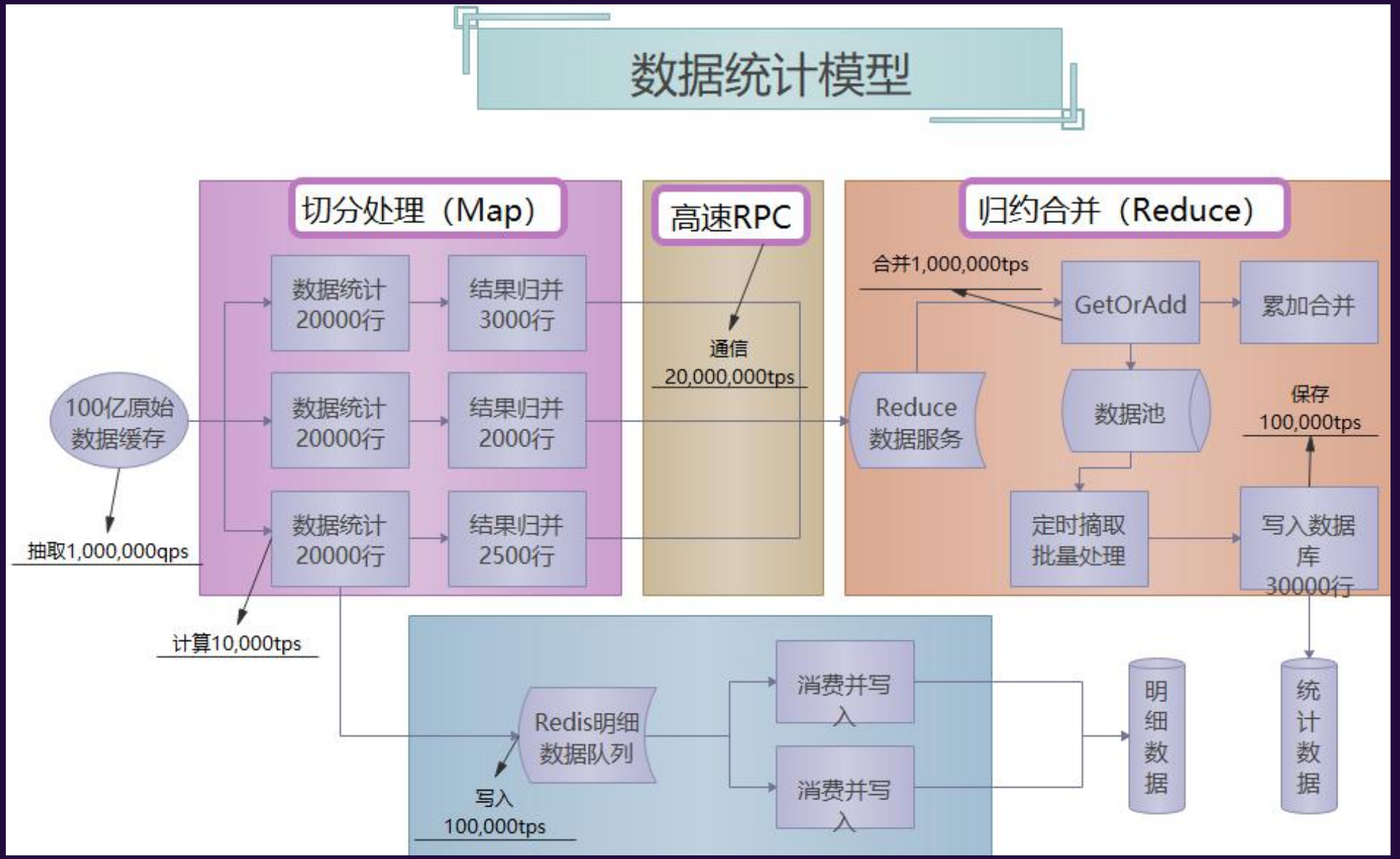
Redis日常17万ops
平均延迟300us
单点压测峰值222万ops
RPC单点压测峰值2256万
tps

系统架构

前中后三套系统分别偏移30秒/540秒/3600秒抽取扫描数据写入Redis，缓存30天，另有两套写备用Redis，缓存10天。

* 以上数据采样于2018年9月

数据统计模型


数据统计模型
切分处理（Map）
高速RPC
归约合并（Reduce）
100亿原始数据缓存
抽取1,000,000qps
数据统计20000行
结果归并3000行
数据统计20000行
结果归并2000行
数据统计20000行
结果归并2500行
计算10,000tps
通信20,000,000tps
合并1,000,000tps
GetOrAdd
累加合并
Reduce数据服务
数据池
保存100,000tps
定时摘取批量处理
写入数据库30000行
Redis明细数据队列
写入100,000tps
消费并写入
消费并写入
明细数据
统计数据

开源项目

- ✓ NewLife.Redis
- ✓ NewLife.RocketDB
- ✓ NewLife.XCode
- ✓ AntJob

```
4   namespace HelloWork
5   {
6       internal class HelloJob : Job
7       {
8           public HelloJob()
9           {
10              // 今天零点开始，每5分钟一次
11              var job = Model;
12              job.Start = DateTime.Today;
13              job.Step = 5 * 60;
14          }
15
16          protected override Int32 Execute(JobContext ctx)
17          {
18              // 当前任务时间
19              var time = ctx.Task.Start;
20              WriteLog("新生命蚂蚁调度系统！当前任务时间：{0}", time);
21
22              // 成功处理数据量
23              return 1;
24          }
25      }
26  }
```

03 Part

项目实战

项目实战

数据清洗

- ✓ 这是一个按照插入时间抽取数据的案例
- ✓ 每张表一个作业，按自己的结构写入Redis
- ✓ 偏移45秒很短，实时性好，但可能漏数据
- ✓ 冗余系统使用2小时偏移抽取，补缺补漏

调度模式：全部

开始时间	结束	步进	批大小	偏移	并行	总数	成功	错误	次数	速度
10:11:39		00:00:05	5,000	00:00:45	8	2,909,352,890	2,749,503,364	0	1,411,997	30,059
10:11:38		00:00:05	5,000	00:00:45	8	141,400,443	138,297,171	0	1,364,343	24,579
10:11:36		00:01:00	5,000	00:00:45	8	31,300,065	31,217,354	0	112,469	23,689
10:11:39		00:00:05	5,000	00:00:45	8	5,975,298	5,710,837	0	1,363,190	23,706
10:11:35		00:01:00	5,000	00:00:45	8	4,278,109	4,205,275	0	112,608	25,436
10:11:00		00:01:00	5,000	00:00:45	8	4,294,232	4,262,193	0	112,442	18,471
10:11:00		00:01:00	5,000	00:00:45	8	3,369,477	3,319,314	0	112,510	18,350
10:10:48		00:01:00	5,000	00:00:45	8	106,617,641	106,288,482	0	112,984	26,282
10:11:33		00:01:00	5,000	00:00:45	8	247,889,137	246,669,271	1	114,175	26,893
10:11:39		00:00:05	5,000	00:00:45	8	2,026,274,619	1,915,512,568	0	1,410,433	27,315
10:11:41		00:00:05	5,000	00:00:45	8	2,848,687,999	2,654,941,147	0	1,418,066	24,545
10:11:42		00:00:05	5,000	00:00:45	8	3,032,100,203	2,834,248,386	1	1,412,882	29,043
10:11:42		00:00:05	5,000	00:00:45	8	5,618,003,144	5,278,427,505	1	1,462,728	33,654
10:11:40		00:00:05	5,000	00:00:45	8	7,157,650,715	6,644,267,915	12	1,491,795	25,760
10:11:42		00:00:05	5,000	00:00:45	8	71,566,760	69,826,119	0	1,364,728	21,856
10:11:40		00:00:05	5,000	00:00:45	8	3,270,002,214	3,057,210,345	6	1,421,902	27,232

数据计算

每个作业会在多服务器节点上进行多线程计算，各自处理不同的时间分片，最后汇总数据

开始	客户端	行	步进	批大小	总数	成功	错误	次数	速度	抽取速度	耗时	全耗时	状态	
2019-09-06 10:12:50	192.168. ..)@6004	0	5	5,000	0	0	0	0	0	0	00:00:00	00:00:00	就绪	
2019-09-06 10:12:45	192.168. ..)@6004	3,649	5	5,000	3,649	3,649	0	1	3,414	73,653	00:00:01	00:00:01	完成	78
2019-09-06 10:12:40	192.168. ..)@6004	4,424	5	5,000	4,424	4,424	0	1	3,264	77,666	00:00:01	00:00:01	完成	78
2019-09-06 10:12:35	192.168. ..)@6004	3,741	5	5,000	3,741	3,741	0	1	3,527	75,559	00:00:01	00:00:01	完成	75
2019-09-06 10:12:30	192.168. ..)@6004	3,978	5	5,000	3,978	3,978	0	1	3,414	75,244	00:00:01	00:00:01	完成	78
2019-09-06 10:12:25	192.168. ..)@6004	3,203	5	5,000	3,203	3,203	0	1	3,670	76,173	00:00:00	00:00:00	完成	78
2019-09-06 10:12:20	192.168. ..)@6004	4,169	5	5,000	4,169	4,169	0	1	3,225	63,157	00:00:01	00:00:02	完成	75
2019-09-06 10:12:15	192.168. ..)@6004	5,282	5	5,000	5,282	5,282	0	1	3,156	61,465	00:00:01	00:00:01	完成	75
2019-09-06 10:12:10	192.168. ..)@6004	2,181	5	5,000	2,181	2,181	0	1	3,062	75,388	00:00:00	00:00:01	完成	73
2019-09-06 10:12:05	192.168. ..)@6004	4,082	5	5,000	4,082	4,082	0	1	3,291	77,762	00:00:01	00:00:01	完成	78
2019-09-06 10:12:00	192.168. ..)@6004	3,573	5	5,000	3,573	3,573	0	1	3,045	77,097	00:00:01	00:00:01	完成	73
2019-09-06 10:11:55	192.168. ..)@6004	4,195	5	5,000	4,195	4,195	0	1	3,101	65,685	00:00:01	00:00:01	完成	75
2019-09-06 10:11:50	192.168. ..)@6004	3,368	5	5,000	3,368	3,368	0	1	3,271	72,393	00:00:01	00:00:01	完成	73

显示名	开始时间	结束	步进
9，账单结算	2019-09-03 07:00:00		1.00:00:00
8，复制旧版统计数据	2019-06-06 23:59:59		1.00:00:00
7，生产刷明细任务	2019-09-07 04:35:00		1.00:00:00
7，更新T-1明细	UpdateDetail/0		
6，更新乡镇网点统计	Static/0		
6，更新一级网点统计	Static2/0		
5，生产T-x统计任务	2019-09-07 05:14:00		1.00:00:00
5，每日统计	2019-09-07 05:13:00		1.00:00:00
5，每小时统计T1/T2/T3	17:00:00		01:00:00
5，每小时统计	16:44:00		00:10:00

汇总统计

- ✓ 数据按批计算后，在数据库或Redis中写入明细表数据，并对统计表进行增量累加，实时性好
- ✓ 定时汇总任务从明细表执行聚合查询，修正统计表数据
- ✓ 每天凌晨重算一次明细数据，更准确
- ✓ 后置定时任务修正几天前统计数据，以免因明细数据改变导致统计数据不准确
- ✓ 重大汇总任务，采用消息驱动模式，分摊到不同节点进行处理

项目实战

分布式

- ✓ 作业分布在多节点上进行多线程处理
- ✓ 每线程处理能力超过1000tps
- ✓ 实际经验，不需要太多节点和线程，瓶颈在数据库写入

进	批大小	总数	成功	错误	次数	速度	抽取速度	耗时	全耗时	状态
5	5,000	0	0	0	0	0	0	00:00:00	00:00:00	就绪
5	5,000	4,328	4,328	0	1	3,402	77,005	00:00:01	00:00:02	完成
5	5,000	2,668	2,668	0	1	3,208	79,719	00:00:00	00:00:01	完成
5	5,000	3,728	3,728	0	1	3,413	66,869	00:00:01	00:00:01	完成
5	5,000	4,285	4,285	0	1	3,194	57,240	00:00:01	00:00:01	完成
5	5,000	4,090	4,090	0	1	3,086	65,967	00:00:01	00:00:01	完成
5	5,000	3,561	3,561	0	1	3,008	76,176	00:00:01	00:00:01	完成
5	5,000	3,028	3,028	0	1	3,447	57,310	00:00:00	00:00:01	完成
5	5,000	4,392	4,392	0	1	3,253	73,723	00:00:01	00:00:01	完成
5	5,000	3,589	3,589	0	1	3,281	77,553	00:00:01	00:00:01	完成
5	5,000	3,489	3,489	0	1	3,024	70,727	00:00:01	00:00:01	完成
5	5,000	4,395	4,395	0	1	3,408	70,250	00:00:01	00:00:01	完成
5	5,000	3,828	0	0	1	6,237	66,545	00:00:00	00:00:01	完成
5	5,000	3,828	3,828	0	1	3,144	70,946	00:00:01	00:00:01	完成
5	5,000	4,539	4,539	0	1	3,124	80,226	00:00:01	00:00:01	完成
5	5,000	3,098	1	0	1	6,448	64,372	00:00:00	00:00:00	完成
5	5,000	3,098	3,098	0	1	3,190	72,410	00:00:01	00:00:01	完成
5	5,000	5,218	5,218	0	1	3,459	59,817	00:00:01	00:00:02	完成

总结
SUMMARY

快递物流行业场景

系统架构设计思想

项目实战落地

谢谢观看