Spark

> 合作分工如下
>
> - 武德钰：2.1~2.4
> - 周琪丰：2.5~2.7
> - 雷镇豪：1、3

一、项目的基本背景和发展历程介绍

1.1 项目背景与技术类型

Spark在2009年作为一个研究项目在加州大学伯克利分校RAD实验室诞生，为什么会有Spark呢？因为研究室的人员当时都使用过MapReduce，他们发现MapReduce操作过于简单（只能通过map,Reduce),对于处理复杂的程序，实现起来很麻烦，并且在迭代计算和交互式计算中效率低下，因此Spark从一开始设计的核心就是迭代算法设计和交互式查询，同时还支持内存式存储和高效的容错机制。Spark于2010年3月开源，并且在2013年6月交给了Apache基金会，现在已经成为Apache开源基金会的顶级项目。综合而言，Spark 是一种计算框架，是用来实现快速而通用的集群计算的平台，它适用于各种各样原本需要多种不同的分布式平台的场景，包括批处理、迭代计算、交互式查询、流处理，通过在一个统一的框架下支持这些不同的计算，Spark使我们可以简单而低耗地把各种处理流程整合在一起。

1.2 版本发布历史

- 2009年，Spark诞生于伯克利大学AMP Lab，属于伯克利大学的研究性项目；
- 2010 年，通过BSD 许可协议正式对外开源发布；
- 2012年，Spark第一篇论文发布，第一个正式版（Spark 0.6.0）发布；
- 2013年，成为了Aparch基金项目；发布Spark Streaming、Spark Mllib（机器学习）、Shark（Spark on Hadoop）；
- 2014 年，Spark 成为 Apache 的顶级项目； 5 月底 Spark1.0.0 发布；发布 Spark Graphx（图计算）、Spark SQL代替Shark；
- 2015年，推出DataFrame（大数据分析）；2015年至今，Spark在国内IT行业变得愈发火爆，大量的公司开始重点部署或者使用Spark来替代MapReduce、Hive、Storm等传统的大数据计算框架；
- 2016年，推出dataset（更强的数据分析手段）；
- 2017年，structured streaming 发布；
- 2018年，Spark2.4.0发布，成为全球最大的开源项目。

1.3 主要贡献者

十年前，Spark在UC Berkeley Amp Lab创立【主作者 Dr. Matei】，然后被捐献给了Apache Software Foundation。现在的Spark社区是由有一群committers和一千多个来自全世界的contributors合力创建。而且就在不久前，网易数帆-有数技术专家燕青接受 Apache Spark PMC 邀请，正式成为 Apache Spark Committer（核心贡献者）。现在给出全部主要贡献者的粗略信息：

- 数据砖厂【Databricks】Matei大神带领的22位年轻小伙伴
- 美国计算机名校群 【UC Berkeley 3位，University of Virginia，University of Wisconsin， Princeton University，University of Michigan，Stanford University 各1位】
- FLAG【Facebook 3位，Microsoft + LinkedIn 1位，Apple 2位，Google 3位】
- 其它IT巨头们【IBM + Redhat 5位，Intel 2位，Oracle 1位，NTT 2位，Nvidia 1位，Yahoo 1位】
- 中国三大云商【阿里巴巴，腾讯 ，华为 各1位】

- 大数据公司【Palantir 1位，Cloudera 3位，Hortonworks 3位，Alluxio 1位，Juicedata 1位】
- 其它Start up 【Uber，Remix，LightStep，Nexstar， ClearStory 各1位】

1.4 CI/CD 的使用

CI/CD是一种通过在应用开发阶段引入自动化来频繁向客户交付应用的方法。CI/CD的核心概念是持续集成、持续交付和持续部署。具体来说，CI/CD可让持续自动化和持续监控贯穿于应用的整个生命周期（从集成和测试阶段，到交付和部署）。这些关联的事务统称为“CI/CD管道”，由开发和运维团队协同支持。它的好处有以下两种：快速发现错误。每完成一点更新，就集成到主干，可以快速发现错误，定位错误也比较容易；防止分支大幅偏离主干。如果不是经常集成，主干又在不断更新，会导致以后集成的难度变大，甚至难以集成。

1.5 其他有价值的信息

Apache Spark作为顶级的开源项目，在世界上也有不小的知名度，它在大数据行业中的首要重要性是因为它的内存中数据处理功能使其与MapReduce相比具有高速数据处理引擎。

Apache Spark具有巨大的潜力，可以为行业中与大数据相关的业务做出贡献。它带来的不同业务优势主要有一下四点：

- 对于专注于物联网的公司来说，这是一个理想的工具。由于其低延迟的内存中数据处理功能，Spark可以应对许多分析挑战。除此之外，它还具有用于机器学习和图形分析算法的完善库。
- 凭借其用于数据流，机器学习，SQL查询，图形分析的高级库，Spark帮助大数据科学家轻松创建复杂的工作流程。这不仅可以减少编码，而且可以更快地洞悉组织的大数据分析。
- Spark可以在现有的Hadoop分布式文件系统（HDFS）上运行，并且可以与Hadoop很好地协同工作。因此，组织无需为Spark建立新的设置。使用相同的数据和群集，他们可以在同一Hadoop群集上部署Spark。对于组织而言，这是一个更加明显的节省成本的增强功能。
- 由于Spark与许多编程语言（例如Java，Scala，Python，R等）兼容，因此易于使用并且需要较少的编码。此外，Spark有大量的程序员社区。因此，组织无需分别租用昂贵的资源。

二、项目的历史轨迹分析

2.1 每月新增 Star 和 Frok 的个数

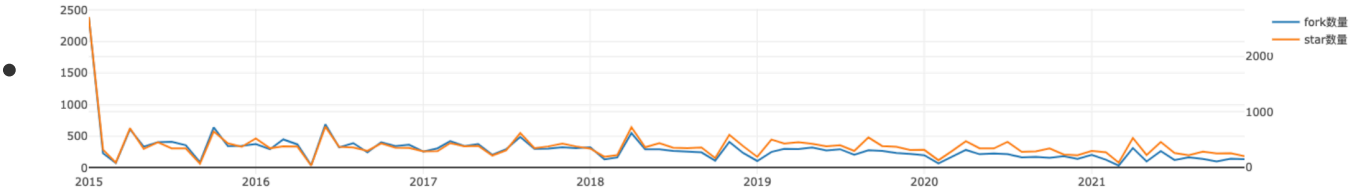
- 每月新增Star的个数

	2015	2016	2017	2018	2019	2020	2021
1	2718	517	284	335	284	311	293
2	315	340	288	186	499	123	267
3	81	376	433	216	426	286	81
4	702	369	377	719	451	468	526
5	328	26	384	357	422	339	218
6	447	731	208	432	376	337	454
7	338	365	303	350	396	456	255
8	337	353	615	340	292	275	213
9	61	293	342	353	536	284	279
10	644	427	373	161	380	338	247
11	428	352	423	584	368	225	251
12	366	345	373	378	310	213	194

每月新增Fork的个数

	2015	2016	2017	2018	2019	2020	2021
1	2390	318	259	327	109	200	205
2	232	295	309	135	252	71	128
3	77	451	424	165	300	176	36
4	611	371	346	552	296	284	314
5	336	41	377	293	322	216	101
6	408	688	207	295	277	230	265
7	414	325	295	268	295	217	125
8	259	392	489	256	207	166	166
9	90	245	301	246	278	174	142
10	640	407	305	114	269	161	101
11	344	345	327	412	236	184	145
12	350	365	313	237	221	141	139

每月新增Fork和Star的个数

- 

2.2 每月打开 Issue 和 关闭 Issue 的个数

每月open Issue的个数

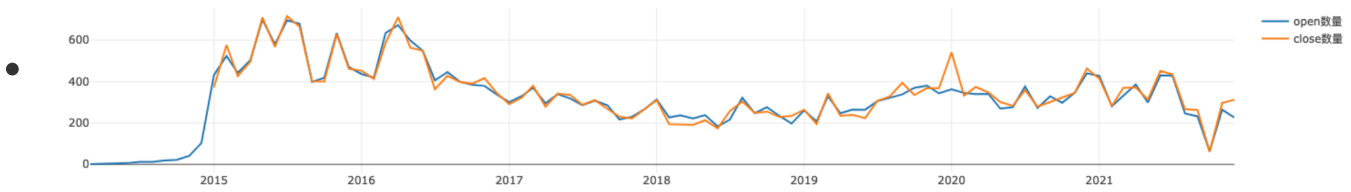
-

	2014	2015	2016	2017	2018	2019	2020	2021
1		432	437	301	313	261	363	427
2		524	419	330	227	207	345	281
3	1	442	635	370	237	329	340	331
4	3	503	673	295	222	247	341	386
5	7	700	599	339	238	265	269	299
6	6	580	549	318	183	264	277	430
7	12	696	406	286	216	305	376	428
8	12	679	446	308	322	322	273	246
9	19	398	399	285	247	338	329	232
10	22	418	385	217	276	370	298	63
11	41	634	379	230	234	380	346	264
12	103	470	338	265	197	343	440	226

每月close Issue的个数

	2015	2016	2017	2018	2019	2020	2021
1	373	453	291	310	264	541	414
2	576	413	322	194	195	333	282
3	427	588	380	194	343	374	370
4	496	711	279	191	235	348	372
5	710	564	342	213	239	301	313
6	568	549	335	174	223	283	451
7	716	363	288	257	306	359	435
8	665	428	310	304	328	279	267
9	401	398	268	248	394	301	263
10	400	390	230	255	335	323	61
11	629	417	222	229	368	346	296
12	462	345	267	234	368	463	312

每月OpenIssue和CloseIssue的个数



2.3 每月打开 PR 和合入 PR 的个数

> 合并PR数据均为null

每月打开PR的个数

	2015	2016	2017	2018	2019	2020	2021
1	433	457	309	324	255	359	417
2	537	440	344	225	205	326	280
3	463	660	389	246	335	336	323
4	517	712	311	222	236	336	389
5	719	608	354	243	260	261	296
6	595	580	322	189	254	264	441
7	705	422	295	229	302	363	435
8	699	471	307	333	312	261	256
9	410	410	300	267	329	312	273
10	438	386	216	274	356	292	44
11	651	383	230	236	372	343	307
12	491	352	275	198	334	427	312

每月打开PR数量

2.4 每月在仓库中活跃（只要有日志产生就算）的不同开发者（也就是一个GitHub账号）总数

每月活跃开发者

Year	2015	2016	2017	2018	2019	2020	2021
1	13297	13565	12483	9253	10279	14846	8809
2	11330	19445	14856	9664	15113	16415	11217
3	15747	21492	8921	10018	10392	15837	13025
4	23299	18207	11875	12393	11634	12706	10169
5	21928	16028	12080	11150	11271	14014	14009
6	29594	13664	9920	12199	15718	21920	14159
7	23312	15109	11053	15119	15188	12610	8798

2.5 Issue 从打开到关闭的平均时长和中位数（单位：天）

- Issue 从打开到关闭的平均时长为 27.58天
- Issue 从打开到关闭的中位数为 13天

2.6 PR从打开到合入的平均时长和中位数（单位：天）

- PR 从打开到合并的平均时长为 0.50天
- PR 从打开到合并的中位数为 1天

2.7 Issue和PR从打开到第一次有人回复（非本人回复）的平均时长和中位数（单位：天）

- PR 从创建到第一次有人回复(非本人)的平均时长为 24.81天
- PR 从创建到第一次有人回复(非本人)的中位数为 2天

三、结合期中分析的归档项目，对比分析活跃/归档项目

3.1 项目基础数据（2.1/2.2/2.3）的变化趋势

1. (2.1)Spark项目在2015年时Fork和Star的数目达到巅峰，接近2500，之后回落至500~1000范围，且持续至今；Faker项目也是在2015年时Fork和Star的数目达到巅峰，超过600，之后回落至50以内的范围，且持续至今；相比之下，Spark项目的Fork和Star规模都是Faker项目难以望其项背的，也可以明显推断出这两个开源项目的规模差异。
2. (2.2)Spark项目在进入2015年前每月打开 Issue 和 关闭 Issue 的个数就持续上涨直至最高点700，然后虽有起伏，但总体波动不大，在2018年有所回落至200后又在2020年反弹至接近600，然后一直在附近持续至今；Faker项目的每月打开 Issue 和 关闭 Issue 的个数也是在进入2015年前就开始上涨，但是至2016年前达到最高峰后开始不断震荡，波动范围较大，且总体的巅峰也没有超越过80，到2021年初，随着Faker项目的归档，这两个数据也归为零值。
3. (2.3)Spark项目从2015年开始每月打开的PR数目一直维持在600左右，在2017年和2020年范围内回落并稳定在200~400之间，在2021年小有回升并维持至今；Faker项目每月打开的PR数目从未超越过40，且波动明显。

3.2 开发者数量（2.4）变化趋势

Spark项目的每月在仓库中活跃（只要有日志产生就算）的不同开发者（也就是一个GitHub账号）总数这一数据，在2015年开始就一路上涨，巅峰甚至接近30k的活跃人数，虽然进入2016年后有所回落，但一直保持在10k~20k的活跃开发者人数范围内，且持续至今，可见其规模；Faker项目在巅峰时期的这一数据也只接近过1000人，且很短暂，2015年至2022年归档前，平均值一直保持在500人左右。

3.3 其他的对比方向

就Spark项目而言，其Issue从打开到关闭的平均时长为 27.58天、PR 从打开到合并的平均时长为 0.50天、PR从创建到第一次有人回复(非本人)的平均时长为24.81天；而Faker项目的这三个数据值分别为：92.94天、 49.64天、126.24天；可见Spark项目的各种响应都明显地比Faker项目要迅速，可以推测问题解决的数量也多得多。

四、项目发展到活跃/归档的主要影响因素及原因

- 对于已归档的Faker项目而言，导致其归档的主要影响因素和归档的原因有两点：
    1. 大多数人只使用1个语言环境，即只需要库大小的一小部分。然而Faker项目一开始就被设计为多语言库，即使只需要 10KB，他们也必须下载 3MB，别无选择。Faker应该是一个没有本地化内容的小型核心库。然后来自世界各地的开发人员将向packagegist贡献本地化包，而无需仓库拥有者监督合并。这样用户只需要下载他们需要的核心和语言环境。但是这个问题在项目归档前，唯一的解决方法就是完全重写，所以这应该是归档的最大的一个原因。
    2. 从Faker作者的博客中了解到，在Faker项目归档前，他已经5年没有写过一行 PHP代码。他有尝试过交出Faker仓库的维护权，但是后续开发者的重写都很难令人满意，进展也非常缓慢，所以基于第1点原因，以及这里对开源社区所有人利益的考量，作者决定将Faker归档。
- 对于目前以及未来可预见地都将持续活跃的Spark项目而言，使其在全世界范围内如此活跃的主要影响因素和原因有以下两点：
    1. 优秀的数据模型和丰富的计算抽象：Spark出现之前，已经有了非常成熟的计算系统MapReduce，并提供高级API(map/reduce)，在集群中运行计算，提供容错，从而实现分布式计算。虽然MapReduce提供了数据访问和计算的抽象，但是数据的重用只是简单地将中间数据写入一个稳定的文件系统(比如HDFS)，所以会产生数据复制备份、磁盘I/O和数据序列化，所以在多个计算中遇到需要重用中间结果的操作时效率会很低。这种操作非常常见，比如迭代计算、交互式数据挖掘、图形计算等等。在认识到这个问题之后，学术界的AMPLab提出了一个新的模型，叫做RDD。RDD是一种容错并行的数据结构(其实可以理解为分布式集合，操作起来就像操作本地集合一样简单)。它允许用户将中间结果数据集显式保存在内存中，并通过控制数据集的分区来优化数据存储处理。与此同时，RDD还提供了丰富的应用编程接口(地图，减少，过滤，foreach，redeceByKey...)来操作数据集。后来，AMPLab在一个叫做Spark的框架中提供并打开了RDD。总之Spark是从MapReduce发展而来的，保留了分布式并行计算的优势，改善了其明显的缺陷。将中间数据存储在内存中提高了运行速度，提供丰富操作数据的API提高了开发速度。
    2. 完美的生态圈——全栈：目前Spark已经发展成包含多个子项目的集合，包括SparkSQL、Spark Streaming、GraphX、MLlib等。其中Spark Core为：实现Spark的基本功能，包括RDD、任务调度、内存管理、错误恢复、与存储系统交互等；sparksql为：Spark用来操作结构化数据的包。有了Spark SQL，我们可以用SQL来操纵数据；Spark Streaming为Spark提供了一个用于流式传输实时数据的组件。提供操作数据流的应用编程接口；Spark MLlib为：提供通用机器学习(ML)功能的库。包括分类、回归、聚类、协同过滤等。它还提供了额外的支持功能，如模型评估和数据导入。GraphX(Graph Computing)为：Spark中用于图形计算的API性能良好，函数和运算符丰富，可以在海量数据上自由运行复杂的图形算法。群集管理器为：Spark旨在高效地将计算从一个计算节点扩展到数千个计算节点。结构化扩展:处理结构化流，统一离线和实时API。