今天突发奇想，看了下我在DolphinDB智臾科技入职的时间，看到是2021年3月3日，算了算距今刚好一年零几天，遂写篇文章，作为对过去一年工作的总结，同时展望一下未来一年的工作计划。在工业界工作，工作内容往往由一个个项目组成，故这篇文章也借一个个我在DolphinDB主导或者参与的几个比较大的项目，来总结我过去这一年的工作。

新的存储引擎TSDB(Time Series DataBase)

新加入公司的第一个项目是与@无敌大饺子一起开发一个新的存储引擎TSDB。这里首先要讲讲为什么要开发一个新的存储引擎，以及这个存储引擎是做什么的。DolphinDB原先是有一款存储引擎的，叫OLAP。OLAP的设计比较简单，基本上就是按列存储，每个文件是表格的一个列，数据写入时的顺序即为数据存储的顺序。这样设计的好处是当列并不是很多的时候写入非常快：只需要在每个列文件上进行追加写即可。同时（当列并不是很多的时候）如果要对表格数据进行大规模的分析（例如策略回测）的效率也比较高。但是随着DolphinDB的应用场景在拓宽，OLAP的存储引擎逐渐碰到了一些问题。其中最大的问题就是在物联网中的点查需求。这里又需要简单介绍一下物联网的时序数据库应用场景是一个怎么样的场景。物联网通常的应用场景都相当简单，会有大量的设备（如几十万，甚至上千万的设备）接入数据库，例如一个省份的电表。每个设备每隔一段（固定的）时间（如一秒）会产生一条数据。**而用户对数据的最主要的需求是去查某个或少数几个设备在给定时间段内的数据**，这样的需求我们称之为点查。在我们试图将DolphinDB应用到物联网的场景时，OLAP存储引擎遇到了困难，这个困难就是点查效率的低下。具体地说，对于OLAP引擎，由于存储数据的顺序即为数据写入的顺序，而对于同一个设备来说，它的数据在写入的时候往往是分散在表格的各个位置的。例如，假设数据库总共有3个设备接入，数据写入的顺序往往是先写入1号设备的第1秒的数据，再写入2、3号设备的第1秒的数据，然后再写入1号设备的第2秒的数据，再写入2、3号设备第2秒的数据，以此类推。很显然，如果数据的存储顺序与写入顺序是一致的，那么假如我们要查询1号设备的数据，就需要遍历整个列文件才能找到所有1号设备的数据，这样的效率是相当低下的。设想如果有1000万个设备，而查询1个设备的数据就需要访问所有的数据，这个效率自然是不会高的。当然，实际情况没有这么糟，DolphinDB有完善的分布式表分区机制（具体可以可以参考这篇文章），使得一个分区的设备数不会太多，但即便如此，在OLAP引擎中进行点查，也往往需要几百毫秒的延时。而TSDB就是一款我们为了解决这个核心的点查问题而设计出来的存储引擎，我们期望它的点查效率能够低至几十毫秒，甚至是几个毫秒。

TSDB的设计是我和@无敌大饺子以及公司的两位领导Davis和大飞哥一起讨论得出的。设计的思路也不复杂：**既然OLAP中点查效率的低下是由于同一个设备的数据是散落在文件四处的，那简单，我们TSDB中就将同一个设备的数据连续存储在一块就可以了。**这很自然地就需要将写入数据进行排序，在结合其他的一些实际需求，我们很自然地引用了LSMT(Log Structured Merge Tree)架构作为TSDB的设计基础。关于TSDB设计的详细介绍可以参考我和@无敌大饺子之前的一个直播录像，我这里就不展开介绍了。在设计完之后，我们进行了任务分工，我负责存储引擎的读取、计算部分，@无敌大饺子负责存储引擎数据的写入部分，大飞哥负责存储引擎的WAL和recovery模块。我们大约花了一个月的时间给出了第一个内部测试的beta版本，后面又陆陆续续地花了一些时间来加入一些新的功能如数据自动去重、增加列、支持blob类型，以及修复系统中存在的bug。在我加入公司的第三个月，我们终于决定将这个存储引擎推出供客户使用。

TSDB这个项目对我来说意义非凡。首先，这是我第一个参与甚至是主导的数据库相关的项目，我借此机会学了众多数据库的知识。其次，这是我第一次参与的复杂的大型项目。之前在前东家也有进行过交易系统的开发，但说实话，那样的项目我觉得称不上大型与复杂，虽然也有很多的代码量，但主要也就是业务逻辑的堆积。TSDB不一样，它中间有大量的状态需要维护，完完全全是一个复杂的系统，想要让它稳定的运行并不容易。而且TSDB大量地涉及到与计算机系统底层打交道，而且我负责的是数据查询与计算部分，性能的重要性不言而喻，我借此机会学会了诸多的性能优化技巧，如leaky buffer, 尽量减少内存拷贝，合理规划读盘的顺序，避免多线程对磁头的争抢等等等等。最后，我们在一个真实的物联网数据集上进行了TSDB与其他数据库的性能对比评测。这个数据集有两百万个设备，我们使用了三台机器组成的集群来进行评测，主要对比了物联网领域常用的数据库在点查方面与TSDB的性能比较。下图给出我们的实验结果：

	线程数，并发任务数	dolphindb用时(ms)		tdengine用时(ms)		clickhouse用时(ms)	
		avg	95%	avg	95%	avg	95%
无缓存	48/10	11	15	30	34	14	15.2
	48/20	13	22	28	32	17.1	19.5
	48/48	17	27	30	35	30.8	36.4
	96/48	19	31	30	37	32.4	39.2
	96/96	30	48	56	62	59.7	70.5
有缓存	48/10	5	7	27	31	13.1	14.8
	48/20	5	8	28	35	16.5	20.8
	48/48	8	12	35	56	25.7	30.2
	96/48	7	10	30	37	25.2	31.9
	96/96	14	19	53	63	52.1	69.5

可以看到，TSDB的点查性能显著优于在物联网中以点查性能著称的ClickHouse与TDEngine。能够做到这一点，毫无疑问TSDB是一个成功的项目，也让我为之自豪。

分布式join

这是我独立完成的一个项目。具体地说，之前的DolphinDB在分布式join方面只支持同数据库下相同分区方式的分布式表做join，且join的列必须全部为分区列。这显然是有较大局限性的。我的方案也很简单，就是对数据进行re-shuffle(或称re-hash)。若join列为左表的分区列，则我们将右表的数据进行re-shuffle，使得re-shuffle后的数据也按join列进行分区，且将左表与右表相同分区方式的子表存储于同一节点上，这样就可以实现将整个分布式join分解为每个节点上的join，最后再将join的结果合并到一块去。类似的，若join列为右表的分区列，则我们将左表的数据进行re-shuffle。倘若join列既不是左表的分区列，又不是右表的分区列，则这是最糟情况，这种情况下会涉及到大量的数据移动，需要将左表和右表的数据都按join列进行re-shuffle，再进行join操作。

这个项目从设计到开发完成总共花了大约两周的时间，作为第一个我独立完成的项目，我还是比较满意的。

在线机器学习(online machine learning)

这是一个我至今尚未完成的项目。这个项目的背景大概是这样的：现在机器学习非常火热，应用的场景很多，DolphinDB当然也是支持机器学习模型的训练和预测的。然而，目前DolphinDB仅支持先训练机器学习模型，再部署机器学习模型这一批数据处理的范式，而不支持在线地根据最新的数据来更新模型。所以在线机器学习这个项目自然就是想支持在线地根据最新的数据来更新模型。由于我之前对机器学习和机器学习系统，所以我很快地就给出了这个项目的设计方案。再由于我一直有其他优先级更高的任务要做，所以我一直试图找一位实习生或者正式工的同学来按照我的设计方案完成这个项目。无奈直到今天还没寻到相应的同学。如果有同学对这个项目感兴趣，请一定要联系我>_<

内存OLTP存储引擎

这个项目其实我参与不多。在9月份之前，这个项目是由@无敌大饺子牵头在设计与实现，后来他去MIT读PhD去了，就由我们的实习生同学@river接过了实现的大旗，并完成了大部分的代码编写，其他一些同学如实习生hj同学完成了B+树的代码实现，实习生@olaoo 实现了logging模块等。这个项目的背景很简单，我们的很多金融客户有内存OLTP数据库的需求，如金融的交易场景等，而DolphinDB其他的存储引擎都是OLAP的，对于高频的删除、修改、更新的支持有限。这个项目的最大的复杂度来源于二级索引的支持和MVCC的支持，尤其当这二者结合到一块去之后，事务的处理将会变得非常复杂。其实我本来也想深度参与这个项目的，但无奈个人精力有限，又不愿意加班，所以最后只能是摸鱼划水了一波。目前这个项目的单机版已经初具雏形了，在性能调优阶段；而之后我们将会开始进行这个项目的分布式版本，这则会有新的挑战：现有的2PC等框架都太重了，性能不够优秀，如何高效地支持分布式事务，以及分布式场景下如何支持二级索引等等。对这个项目感兴趣的同学，可以联系我>_<

Python Parser

这个项目实际上我完全没有参与，是由我的好友jc与公司CEO/CTO Davis一直在牵头设计与实现的。具体地说，为了提供更强大更灵活的计算能力，DolphinDB提供了一门自带的编程语言DolphinDB Script，其语法类似于Python与SQL的结合。使用这门编程语言，用户就可以在数据库中方便而高效地完成各种复杂的计算操作。然而，语法类似Python，也终究不是Python，用户仍然有一定的学习成本；更重要的是，在数据分析领域，Python的两大重量级的包numpy和pandas早已深入人心，大家已经习惯了它们的操作语法与方便性。Python Parser这个项目，**目标就是让DolphinDB原生地支持使用Python来管理存储与计算，这样用户的学习成本就可以降至最低，而且Python Parser还将原生地支持numpy和pandas的操作。**具体地说，我们自行在DolphinDB内部实现了一个Python的解释器，而且这个解释器将没有GIL的束缚，计算性能会非常出色。我们将在今年三月底四月初推出第一个beta版本的Python Parser，但之后仍然有非常多的工作需要做。对这个项目感兴趣的同学，可以联系我>_<

总结完了过去一年里我参与、主导的一系列较大的项目，下面我将简单介绍我们未来一段时间内打算做的项目，欢迎对这些项目感兴趣的同学加入我们。大家也可以看到，在我们公司，即便是实习生也可以承担重任，所以不用担心自己资历不深而无法参与到核心的项目~

分布式内存OLTP数据库

这个在前面我已经介绍过了，这里就不详细介绍了。

Python Parser

同样，在前面已经介绍过了，这里就不赘述了。

流式数据库

DolphinDB已经有一个非常强大的流计算引擎了。然而目前的流计算引擎的管理和运维还比较麻烦。之后我们计划实现一个流式数据库，能够方便地进行流计算的管理和运维，并支持自动化的流式复杂指标计算。

基于k8S和容器的自动化部署

目前DolphinDB的部署和升级还比较麻烦。我们目前已经在开发基于k8S和容器的自动化部署，未来希望能够继续强化这项功能。

新硬件的支持

过去的数年内涌现了众多的新的硬件，如NVM, Infinity Band, RDMA, 高性能的GPU等，然而却鲜有将这些新硬件融入到数据库内的。我们计划未来一段时间内探索如何将这些新硬件与数据库结合，以进一步地提高数据库的性能。

对以上任一项目感兴趣的同学都可联系我~

总结

过去的一年里，我在DolphinDB按照自己期望的方式成长了很多。希望在接下来的一年里，我能够继续这样的成长。