# 论文总结

## 背景介绍

本篇论文是2019年USENIX ATC会议发表的一篇论文。该会议寻求高质量的原创文章，以改善和进一步提高计算系统的知识，重点是实现和实验结果。我们对从小型嵌入式移动设备到数据中心和云的各种规模的系统都感兴趣。 USENIX ATC的范围涵盖与系统软件有关的所有实际方面，包括：操作系统；运行时系统；并行和分布式系统；存储;联网;安全和隐私；虚拟化;软硬件交互；绩效评估和工作量表征；可靠性，可用性和可扩展性；能源/电力管理；错误查找，跟踪，分析和故障排除。

## 课题摘要

采用不可变数据抽象的数据分析框架通常为故障恢复和减轻混乱提供更好的支持，而采用可变数据抽象的框架则对迭代工作负载更有效，这归功于它们对就地状态更新和异步执行的支持。现有的大多数框架都采用两种数据抽象中的一种，而没有另一种的好处。

在本文中，我们提出了一种新颖的编程模型MapUpdate，它可以确定分布式数据集在应用程序中是可变的还是不可变的。我们证明了MapUpdate不仅提供了良好的表达能力，而且还使我们能够享受可变和不变的抽象的好处。 MapUpdate自然支持迭代和异步执行，并且可以根据故障情况自适应地使用不同的恢复策略。我们在名为Tangram的系统中实现了MapUpdate，该系统具有新颖的系统设计，例如轻量级本地任务管理，基于分区的进度控制和上下文感知的故障恢复。广泛的实验验证了七巧板在各种工作负载（包括批量处理，图形分析和迭代机器学习）上的优势。

## 主要内容

首先采用不可变数据抽象的代表性系统有MapReduce、Spark等，其优点有丰富的函数API可以调用，使用功能数据流图对数据集之间的依赖关系进行建模，高效的故障恢复，高效的负载平衡，其缺点不适合有状态表示，仅支持BSP（同步并行），采用可变抽象的系统有以顶点为中心的图形分析系统，例如Pregel、GraphLab、PowerGraphics等，以及基于参数服务器的机器学习系统，例如参数服务器、Petuum等，其优点有高效的迭代工作负载，可能支持异步执行，缺点是需要从最新的检查点完全重新启动（Pregel）或使用昂贵的赋值来进行容错（参数服务器），依赖于应用程序的负载平衡性。

为了可以同时获取不可变抽象以及可变抽象的优点，设计了一种可以区分应用程序中可变部分和不可变部分的编程模型MapUpdate。它提供了一种简单的机制：更新集合是可变的，以及其它集合（如果不同于更新集合），则认为是不可变的。从而让系统从API调用中确定某个几个在集合中是否可变。与MapReduce模型相比较，MapReduce模型，其在数据流抽象中，对集合应用操作病生成新的集合，而MapUpdate模型它让数据集合是可变的，并将减少操作更改为有状态更新操作。



执行MapUpdate命令时，计算机会在其Map集合的本地分区上启动并行Map任务，每个Map任务都会对A的分区中的对象执行map\_func，来生成中间结果，同时map\_func也可以使用集合B中的信息，最后使用update\_func将中间结果交给集合C中的相应对象。

MapUpdate编程模型有三个特征：

（1）部分或全部Map集合（A）、侧输入集合（B）、更新集合（C）可以是相同的集合。

（2）固有的支持迭代和异步执行。

（3）一种简单的机制，用于确定集合在MapUpdate计划中是否是可变的。

根据这三个特征可以将MapUpdate应用到不同的应用中，例如批量处理、迭代机器学习、顶点中心图分析、分布式搜寻器以及流水线工作量：

**（1）批量处理**

MapUpdate可以轻松实现MapReduce所针对的批量处理工作负载，这些负载通常是无状态的，非迭代的，并且仅涉及批量数据移动。 我们通过单词计数示例进行说明，该示例类似于Spark中的示例：map函数在扫描本地文档时生成（单词，计数）对，而update函数将（单词，计数）对汇总为最终计数。 请注意，在单词计数计划中，没有侧面输入集合。

**（2）迭代机器学习**

迭代机器学习算法重复定义一组模型参数，并从训练样本中计算出更新。这些算法（例如，SGD）通常对于异步执行是健壮的，其中使用过时或不一致的模型参数来计算更新。基于参数的系统（例如，Parameter Server [31]，PetuumBösen[60]）被广泛用于分布式机器学习，并支持从异步执行中受益的SSP和ASP。七巧板可以通过将模型参数用作侧输入集合和更新集合来对参数服务器进行建模。我们展示了使用SGD（s = 2）的SGD训练逻辑回归的示例。映射函数使用模型参数计算局部样本的随机梯度，而更新函数将梯度更新提交给模型参数。可以使用表2中的setIter和setStalenss命令来配置迭代和异步执行。请注意，当未配置setStalenss时，Tangram默认使用BSP。

**（3）顶点中心图分析**

以顶点为中心的图形分析系统（例如，Pregel [36]，PowerGraph [21]）通常根据相邻顶点的状态迭代地更新顶点状态。 七巧板可以通过将顶点状态集合用作地图集合和更新集合5来对以顶点为中心的图形处理进行建模。我们以PageRank为例。 贴图函数计算顶点的PageRank值对其邻域的贡献，而更新函数则合并邻域的贡献。 将等级和链接集合进行共分区（通过使用同一分区程序），以减少通信开销。 类似地，七巧板还可以实现以边缘为中心的模型[51]。该模型可以通过共同分区（通过使用相同的分区程序）来减少通信开销。 同样，七巧板也可以实现边缘中心模型[51]。

**（4）分布式搜寻器**

七巧板通过使用URL作为地图集合和更新集合来支持搜寻器。 map函数下载当前URL指向的网页并提取新的URL，而update函数将新的URL插入url集合并将已处理的URL标记为已访问。 请注意，没有侧面输入集合。 setIter（-1）继续执行迭代（即继续爬网），而setStaleness（-1）表示使用ASP。

在上面的应用程序中，我们使用三个集合（A，B，C）的不同组合来实现不同的计算模式。Tangram还支持许多其他在现有系统中难以实现的应用程序（例如Nomad [69]和图匹配[12]）。

**（5）流水线工作量**

MapUpdate对于管道式工作负载特别有用。 实际上，Tangram项目是由诸如阿里巴巴这样的公司中常见的生产数据分析工作负载驱动的，该工作负载由涉及不同类型任务的管道组成。 典型的管道以MapReduce样式的数据处理开始，然后进行各种高级分析（例如，参数服务器样式的模型训练），最后以测试和验证结束。 我们仅以用户分类管道和欺诈检测管道为例进行描述。

在欺诈检测管道中，目标是找到使用虚假交易来破坏规模记录的恶意卖方[47]。 首先处理用户（即买者或卖者）与动态支付活动之间的静态关系，并从预处理数据中提取图形以建模买卖双方的互动。然后，将图匹配应用于与某些与欺诈模式相对应的预定义模板相匹配的交易模式。 图书馆。 验证过程通常涉及MapReduce（例如，加入以获得可疑用户的详细信息的联接）和图形分析，例如计算从可疑用户到黑名单用户的距离。

正如我们将在实验中显示的那样，使用各自的专用系统在管道中处理不同的任务会导致昂贵的上下文切换开销，从而导致系统转储/加载输出/输入数据。 对于单个管道使用多个系统也会损害鲁棒性，因为不同的系统提供不同的容错语义，并要求工程师学习/调整所有系统。 借助具有表达力的MapUpdate API，统一的容错语义和高效的功能，Tangram（实现MapUpdate的系统）可以在统一的框架中处理整个管道，从而完全消除了上下文切换的开销。 此外，统一的MapUpdate API还可显着降低开发成本，而无需用户学习许多系统。

本论文最终在Tangram上实现了MapUpdate，Tangram系统采用大师级工人架构。 主机负责DAG调度（协调工作人员执行可运行的计划），进度跟踪（管理进度并从工作人员那里收集执行统计信息，以实现容错和缓和缓解），以及分区管理（通过维护来跟踪分区的位置） 分区图的主副本）。 工作进程充当分区的分布式内存存储，每个工作进程都使用本地控制器来管理本地任务执行。 对于调度，主机仅向工作人员发出控制命令（启动，更新进度，迁移，恢复等），并且本地控制器负责调度其自己的任务。 本地控制器还与分区主机同步分区图的本地副本和执行进度。 这种设计减少了集中式调度开销，对于扩展到大型集群至关重要。

其主要设计有：

1. 本地任务管理
2. 基于分区的进度控制
3. 上下文感知的故障恢复：支持BSP、SSP和ASP执行模型，记录每个分区提交的更新的位图。
4. 使用分区迁移进行散乱缓解
5. 通讯优化

Tangram区分了两种故障场景，即本地故障和全局故障，并应用了不同的故障恢复策略。

本地故障：故障机器不包含更新（可变）分区，会在运行状况良好的计算机上重新加载丢失的分区（不可变），但并不会直接影响运行状况良好的计算机上的任务执行。

全局故障：故障机器包含更新集合的分区。全局故障会影响所有计算机上的计算。此时Tangram会从最新的检查点重新加载丢失的可变集合，并将全局进度和所有分区的进度重置为最新的检查点，对于不可变集合，Tangram仅重新加载丢失的分区。

Tangram：不必选择可变或不变的抽象，而是可以将他们都包含在一个统一的框架中，并且支持异步迭代工作负载，区分故障恢复与负载平衡