Parallel *k*-means on stream data

1st Author

1st author's affiliation  
1st line of address  
2nd line of address  
Telephone number, incl. country code

1st author's email address

2nd Author

2nd author's affiliation  
1st line of address  
2nd line of address  
Telephone number, incl. country code

2nd E-mail

3rd Author

3rd author's affiliation  
1st line of address  
2nd line of address  
Telephone number, incl. country code

3rd E-mail

**Abstract**

CluStream、streamKM++、mini-batch k-means等k-means算法的变种可用于流式环境下，本文分析了这些算法的特征，包括算法的定义、在计算、参数更新和反馈上的特点，总结了算法的并行化情况，提出算法的评价方法，对测试结果进行评价，对算法的设计人员提供算法在流式化和并行化设计上的选择和评价的指导。

# Introduction

k-means算法是最常用的聚类算法之一，广泛应用与文档分类、犯罪地点识别、客户分类、欺诈检测等场景，这些场景中数据往往是流式且非平稳分布的，因此模型需要随着时间变化。

流式数据上，传统批处理的k-means随着训练数据的增多重复训练全部历史数据，导致算法的实时性差。

流式数据上的k-means聚类算法的经典变种包括CluStream、streamKM++、mini-batch k-means等，他们将历史的数据用历史信息参数的形式表达。目前这些算法在StreamDM、SAMOA、Spark Streaming、Angel等批处理和流处理机器学习平台上实现。

本文以基于Spark Streaming实现的StreamDM平台上的CluStream和StreamKM++以及直接基于Sparking Streaming实现的mini-batch kmeans 为例，分析这三个算法的流式化和并行话特征，分析其当前并行化实现方式及优缺点，改进并行化实现，并指导算法设计人员如何设计对应算法的并行化方式。

为了帮助算法设计人员和使用人员能够根据数和应用场景需求选择合适的流式k-means算法，本文总结了算法的特征、并行化方法、对算法进行测试和评价，回答的研究问题包括：

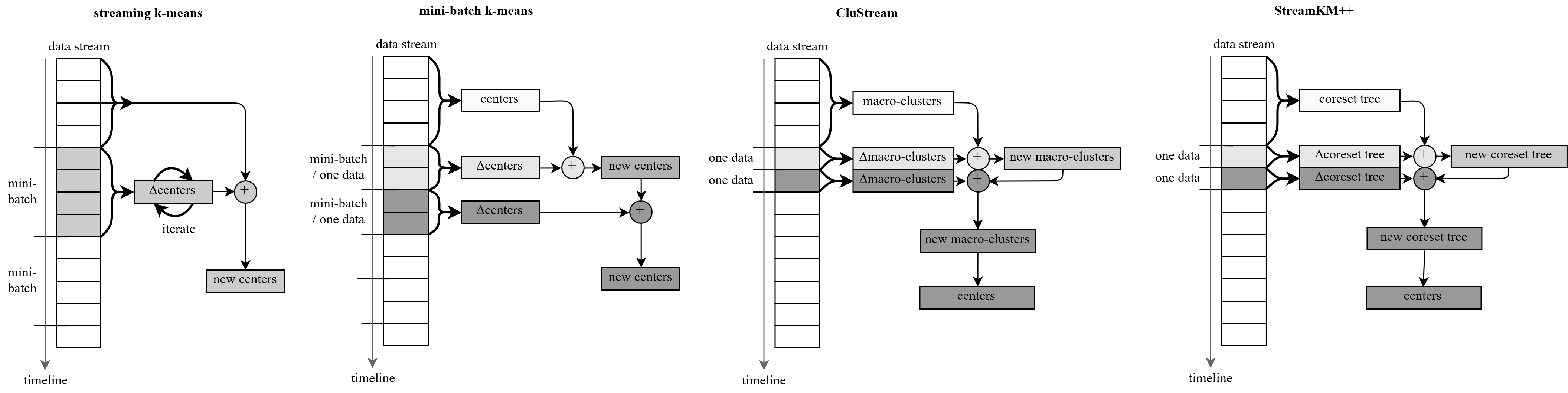


Figure 1

1. 算法分别是什么计算模式？
2. 算法的并行化情况和并行化方法如何？
3. 如何对流式k-means算法进行测试，测试指标包括哪些，如何测试，以及如何评价？

论文贡献：

1. 总结了算法特征、计算模式。
2. 基于算法的现有实现，在spark streaming上并行化实现并改进。
3. 根据算法的特征进行针对性测试，提出算法的评价指标和评价方法。
4. 指导算法设计人员如何根据算法特征、数据特征和场景需求并行化设计算法。

# Background

## 流数据上的*k*-means算法

Table 1

|  |  |  |  |
| --- | --- | --- | --- |
| 算法 | 参数表示 | 计算模式 | 发布时间 |
| Clustream[3] | 微簇 | Online + sketch | 2003 |
| streamKM++[2] | Coreset tree | Online + sketch | 2012 |
| Mini-bath k-means[1]/ Spark Streaming k-means | 模型-聚类中心(centers) | Online | 2010 / 2015 |

算法在计算模式上的对比如figure 1所示。Mini-batch k-means算法属于在线学习算法，本文将在线学习算法的流式实现称为在线增量更新计算模式，每到达一条数据更新一次参数，即历史信息的保存形式，在该算法中，参数为聚类中心和元素计数；Clustream算法为流式数据挖掘算法，本文结合其数据的计算实际和参数的表达方式将该种算法归纳为在线概要更新计算模式，该算法的参数为微簇集合，包含q个微簇，微簇为(2d+3)维的元组，其中q远大于最终的聚类中心数k，d为数据维度；StreamKM++算法同样属于在线概要更新的计算模式，与CluStream算法不同的是，该算法的参数形式称为Core-set tree。

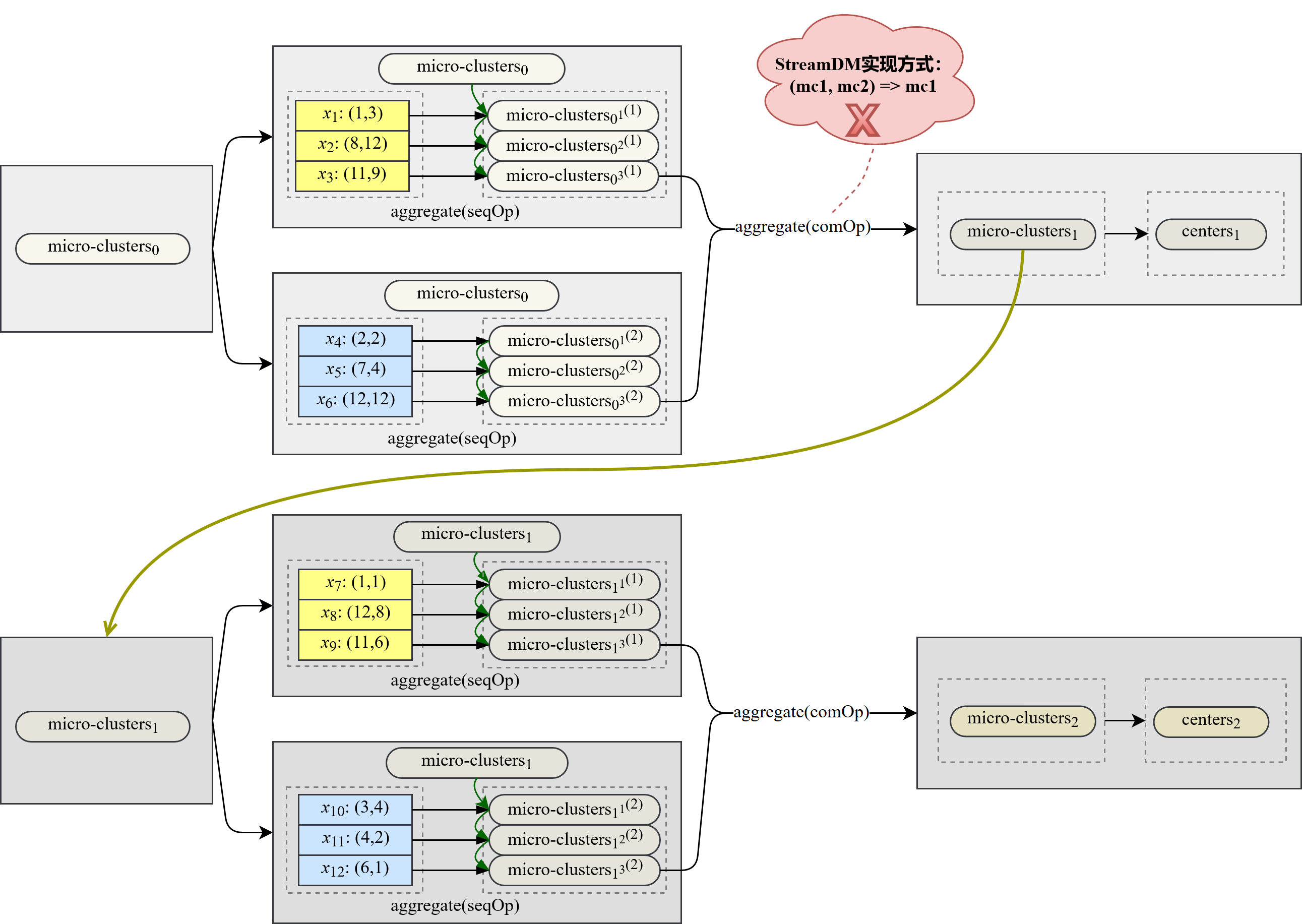


Figure 2 CluStream分布式实现

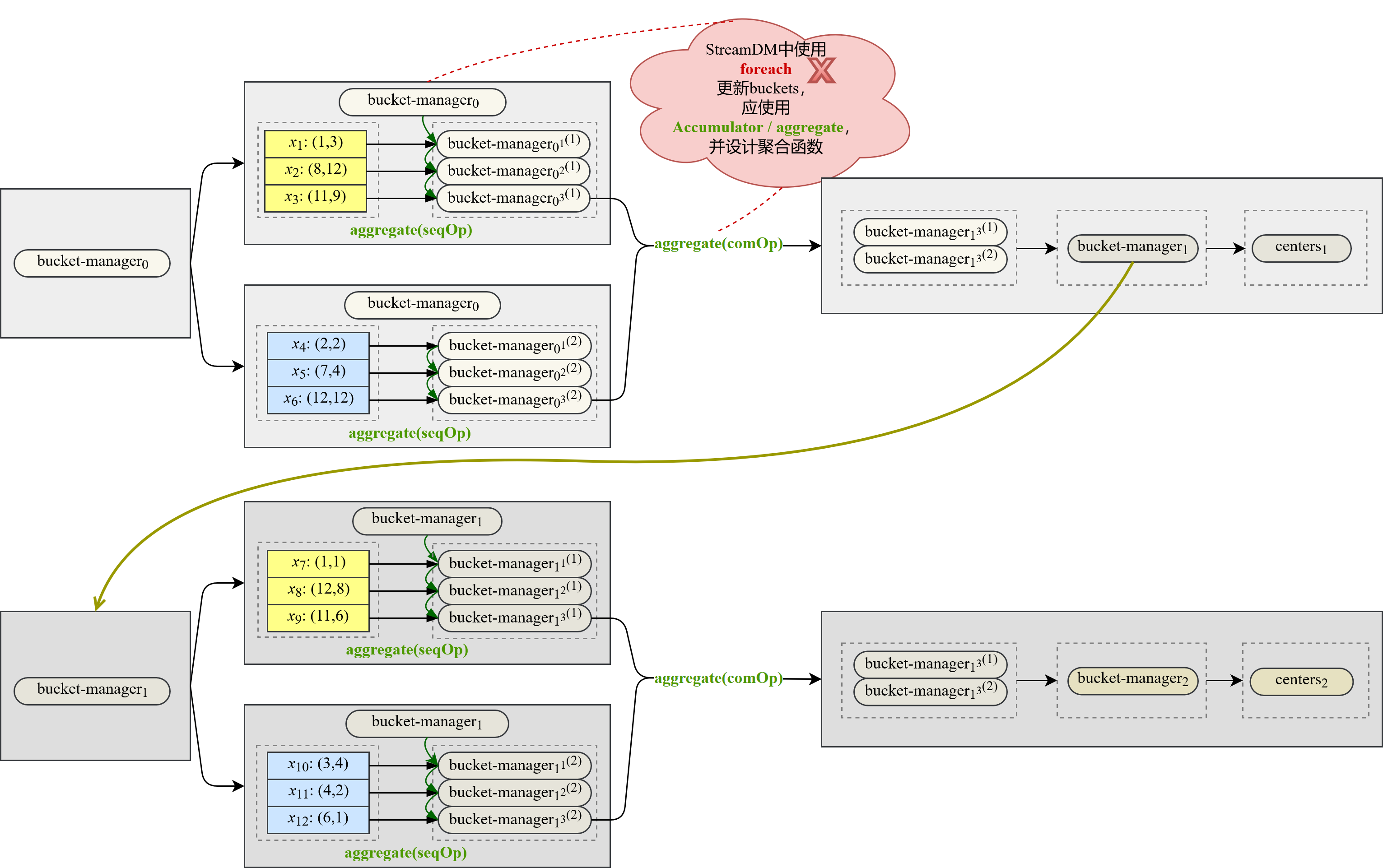


Figure 3 StreamKM++分布式实现

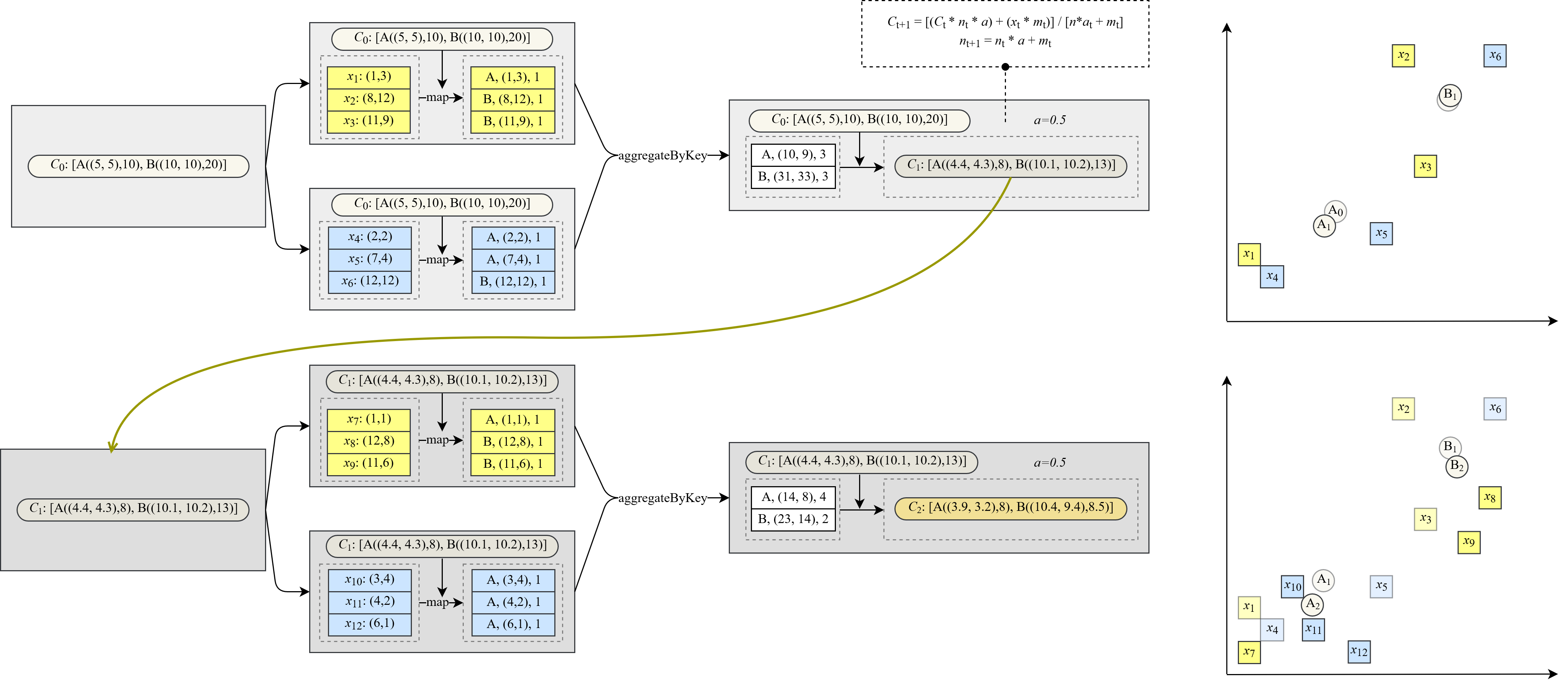


Figure 4 Spark Streaming KMeans分布式实现

## 并行化实现

|  |  |  |
| --- | --- | --- |
| 算法 | 并行方式 | 实现平台 |
| CluStream | 数据并行 | StreamDM |
| StreamKM++ | 数据并行 | StreamDM |
| Mini-batch k-means | 数据并行 | Spark Streaming |
| 数据并行+模型并行 | Angel |

当前并行化实现主要集中在StreamDM平台、Spark Streaming和Angel上，其中StreamDM和Angel都是基于Spark实现的。Mini-batch k-means在Spark Streaming上的并行化流式实现称为streaming k-means。将现有的并行化实现StreamDM上的CluStream、StreamKM++和基于Spark streaming的streaming k-means数据流图表达为Figure 2、Figure 3、Figure 4所示。他们的并行化实现方式都为数据并行，将在线的每到达一条数据计算一次改为一微批数据的并行计算。

然而，StreamDM中实现的CluStream、StreamKM++算法在参数更新和聚合操作上存在错误，其中CluStream算法中没有设计参数的聚合操作，StreamKM++使用foreach算子更新参数。本文更改代码，包括CluStream的aggregate(comOp)，实现原理为每两个微簇集合合并时找到最近的q个微簇；StreamKM++的aggregate，实现原理为aggregate(seqOp)对每个数据更新bucketmanager，aggregate(comOp)结合两个bucketmanager时，用其中一个bucketmanager中的每个元素，更新另一个bucketmanager。

分析一个微批n条数据的算法时间复杂度如下。

CluStream:

aggregate(seqOp) ~

aggregate(comOp) (*p*: #partition)

micro-clusters - cluster (*t*: #iteration)

StreamKM++:

aggregate(seqOp) (*m*: coreset size)

aggregate(comOp)

bucketmanager - cluster

Streaming *k*-means:

空间复杂度如下。

Streaming k-means:

StreamKM++:

CluStream:

算法的参数除去系统参数外，还包含输入数据的参数和算法超参，数据参数和算法超参的配置不同，会影响算法的正确性和性能。

1. 数据参数

|  |  |  |
| --- | --- | --- |
| 参数 | 含义 | 影响 |
| d | 维度 | 见时间复杂度 |
| n | Batch大小 | 实时性 |
| Rate | 流速 | 流处理速度 |
| Sparsity | 稀疏程度 | 计算最近中心时复杂度与稀疏程度关系分析 |

1. 算法超参

CluStream

|  |  |  |
| --- | --- | --- |
| 参数 | 含义 | 取值 |
| k | 聚类中心个数（模型维度） | 依照数据 |
| q | 微簇个数 | 远大于k，与单机内存相关 |
| initial\_buffer\_size | 初始化数据集大小 | 1000（默认） |
| max\_iterations | 最大迭代次数 | 100（默认） |
| Radius | 微簇允许加入新数据的半径阈值 | 2（默认） |
| M | 微簇时间戳标准差，从而进行过时分析 | 100（默认） |

StreamKM++

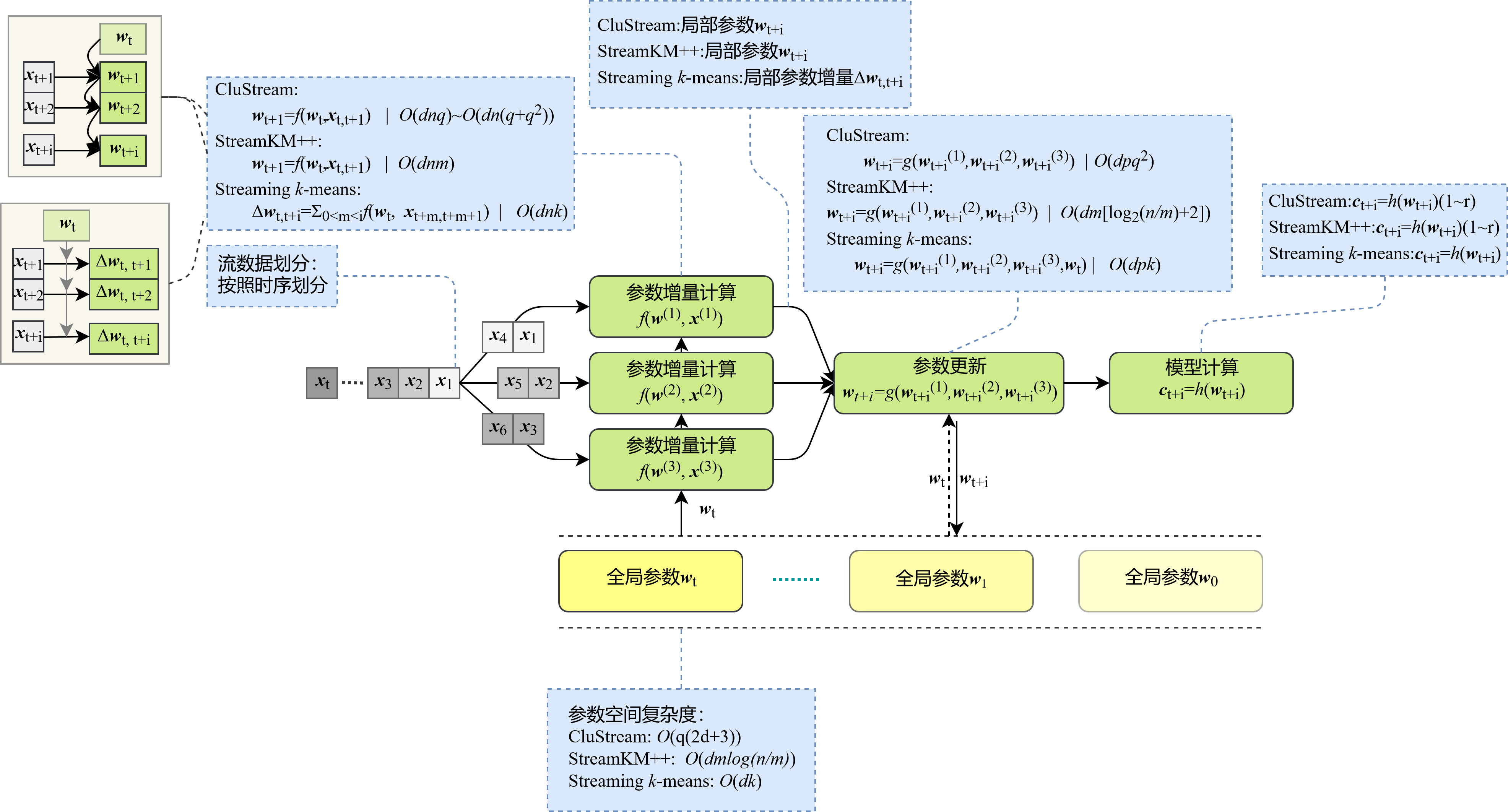
|  |  |  |
| --- | --- | --- |
| 参数 | 含义 | 取值 |
| k | 聚类中心个数（模型维度） | 依照数据 |
| S | Coreset大小 |  |
| Window\_size | Window大小 |  |
| max\_iterations | 最大迭代次数 | 1000（默认） |

Streaming k-means

|  |  |  |
| --- | --- | --- |
| 参数 | 含义 | 取值 |
| k | 聚类中心个数（模型维度） | 依照数据 |
| Decay\_factor | 衰减系数 | 0.5（默认） |
| Weight | 初始化各个簇的权重 | 100（默认） |
| max\_iterations | 最大迭代次数 | 1000（默认） |

# Methodology

## 特征分析



根据上文对三个算法的流式化和并行化的分析，归纳算法的特征。

其中，Batch时间为。

三个算法的计算过程可以归纳为参数增量计算、参数更新和模型计算。

在数据划分方式上，都采用按照时序的数据并行划分方式。

参数增量计算的公式概括为,其中，CluStream和StreamKM++的计算过程中，参数不断更新，下一次计算的参数依据上一个更新完的参数，参数增量计算在微批结束后的输出为最新的参数，在并行化实现中，输出为局部参数；而Streamg k-means中，所有参数增量计算都依据该微批初始化的参数，在微批内参数不作更新，只做增量的聚合，微批结束后输出局部的参数增量。特殊地，Clustream算法的参数增量计算过程复杂度不固定，下限为，上限为。

参数更新的计算公式概括为

,其中，CluStream算法和StreamKM++算法的参数更新过程是聚合各个计算任务上的局部参数，而Streaming k-means算法的参数更新过程是聚合各个计算任务上的局部参数增量，并依据微批前的旧参数，计算新参数。

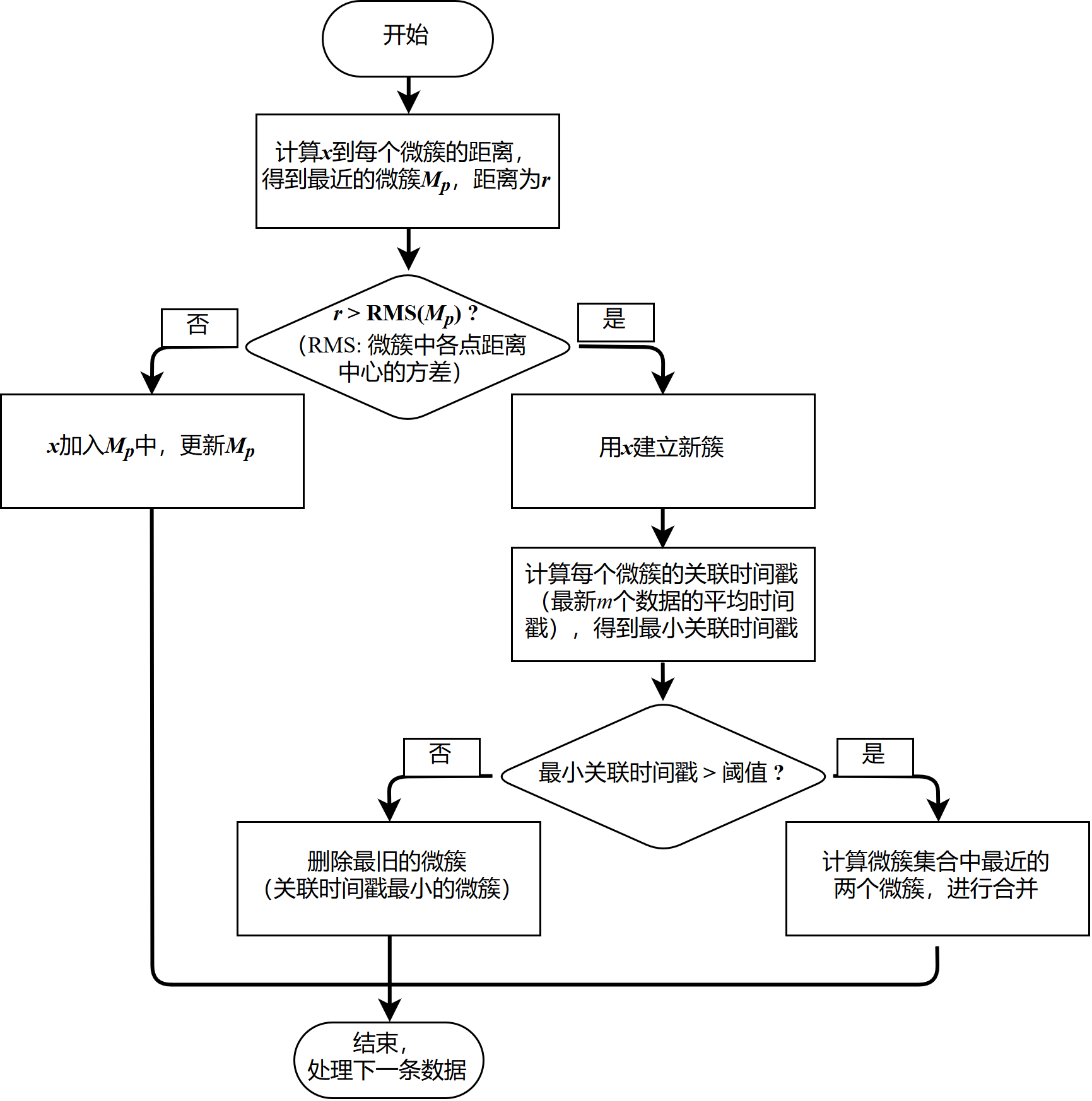
模型计算的计算公式概括为, CluStream算法和StreamKM++算法是在参数上迭代计算得到模型，Streaming k-means算法是做一次计算直接得到模型。

具体的计算步骤的细节如下。

**CluStream**：

1. 参数增量计算

函数的具体过程：计算时机为每一条数据到达计算一次。计算量随着数据的分布改变。



1. 参数更新

函数的具体过程：

每两个长度为的mcs合并成一个长度为的mcs。两两合并，在并行化改进上可以用树模型的层次合并。计算时机为一微批数据输入完毕。

1. 模型计算

**StreamKM++：**

1. 参数增量计算

函数的具体过程：计算时机为每一条数据到达计算一次。

1. 参数更新

函数的具体过程：

合并计算分片上的bm，得到新的bm。具体过程是将一个bm中的每一个元素插入到另一个bm中。计算时机为一微批数据输入完毕。插入过程的时序改变对结果是否有影响？即是否可以用树模型改进？

1. 模型计算

**Streaming *k*-means：**

1. 参数增量计算

计算分片上的数据引起的本地增量。对每一条数据计算一次。

1. 参数更新

聚合各个计算分片上的参数增量，更新参数。

1. 模型计算

## 并行化改进

1. 参数增量聚合的改进

改进StreamDM中的参数增量聚合函数，进一步加速聚合（树型合并）。

2）参数存储

当前实现将参数存储于driver端的内存中，在每一个批次数据计算时，分发到各个计算节点，参数增量计算完成后，合并增量到driver端，更新参数。这种实现方式使用闭包的方式分法参数，当参数过大时，会导致内存溢出问题。

使用mapWithState将参数存储与executor端内存中的实现方式无法更新全局参数。

# Experimental results

## Experimental setup（实验环境）

We performed the evaluation on 9 *mn4.2xlarge* nodes on Alibaba Cloud. One node serves as the master node, and the others are worker nodes. Each node has 4 physical cores (8 virtual cores), 32GB RAM, and 1200GB disk with 3000 IOPS.

## 测试方案

## results

# Discussion

# Related work

# Conclusions

# REFERENCES

1. Sculley D. Web-scale k-means clustering[C]//Proceedings of the 19th international conference on World wide web. ACM, 2010: 1177-1178.
2. Ackermann M R, Märtens M, Raupach C, et al. StreamKM++: A clustering algorithm for data streams[J]. Journal of Experimental Algorithmics (JEA), 2012, 17: 2.4.
3. Aggarwal C C, Philip S Y, Han J, et al. -A Framework for Clustering Evolving Data Stream- s[C]//Proceedings 2003

VLDB Conference. 2003: 81-92