基数估计算法在大数据场景下的应用

夜泓（阿里巴巴商家数据部）

2013-09-14

Outline

个人简介

- 个人信息
  - 本名张洋，花名夜泓
  - 阿里巴巴商家数据部，数据挖掘
- 联系方式
  - 微博：@ 敲代码的张洋
  - 博客：http://blog.codinglabs.org
  - E-mail：ericzhang.buaa@gmail.com

什么是基数

- 定义
  基数又叫做势，是一个可重复集合中 **不重复** 元素的个数。
- 举例

集合	基数
$\{1, 0, 1, 1, 0, 0, 1\}$	2
$\{1, 2, 3, 4, 5\}$	5
N^+	∞
R	∞
$\{0, 0, 0, \cdots\}$	1

什么是基数计算

给定一个 **含有重复元素** 的 **有限** 集合，计算其不重复元素的个数。

互联网应用中的基数计算场景举例

- 计算某个淘宝店铺一天内有多少不同的用户访问
- 计算某个淘宝店铺一天有多少来自北京的女性用户成交
- 某视频网站一月内有多少不同用户观看了美剧频道的视频

基数计算的核心问题

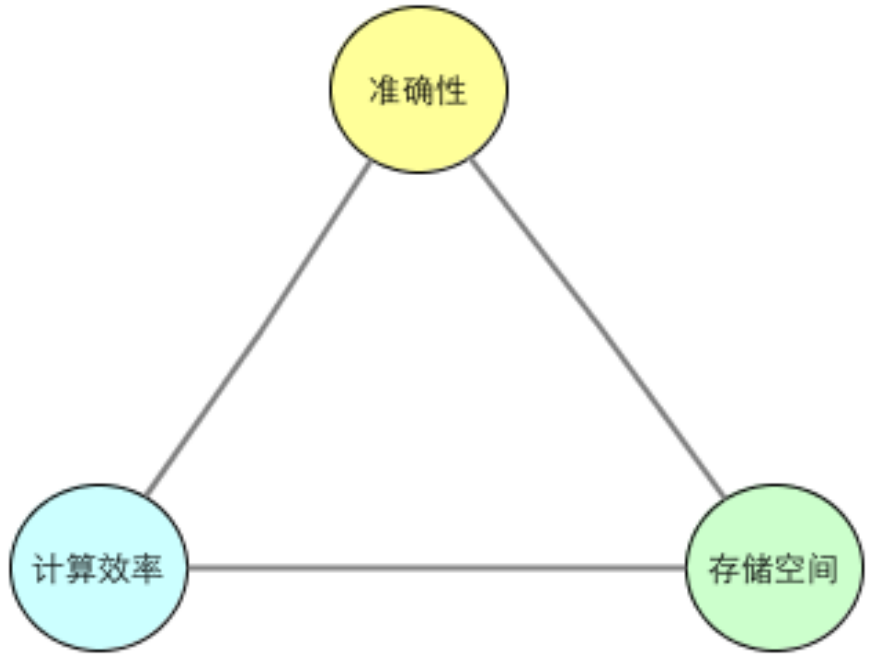


Figure: 基数计算的权衡

常见实现方式

- 离线计算
  - 分布式离线计算(Hadoop ,Hive ,Pig ,...)
- 实时计算
  - B Tree + 计数器
  - bitmap + 计数器

大数据场景下基数计算的困难

- 离线计算
  - 维度爆炸
- 实时计算
  - B Tree 具有不可合并性，无法应对维度爆炸
  - 简单 bitmap 便于合并，但内存开销过大

什么是基数估计算法

- 定义

  基数估计算法是一类概率算法，可以在误差可控的前提下以远低于精确计算的时间和空间消耗对基数进行估计。
- 算法特点
  - 误差可控
  - 时间和空间复杂度仅与估计值标准差及基数上限有关
  - 可合并

基数估计算法的原理

基本原理：以均匀分布的统计理论为基础，寻找便于计算和合并，并且与基数存在随机量化关系的统计量，以此统计量通过量化关系反推基数。

基数估计算法的原理（续）

预处理：由于基数估计算法均以均为分布为前提，因此需要对数据进行均匀化，主要手段为用哈希函数计算原始数据的哈希值，用估计哈希值集合的基数代替估计原始集合的基数。

几点说明

- 基数估计算法分析均假设哈希函数不存在碰撞，因此选用的哈希函数需要哈希空间足够大，且碰撞率低。
- 哈希函数的均匀性要足够好，以满足基数估计算法对于均匀性的需求。
- 实际中常被用来做基数估计哈希的函数有 murmurhash 和 lookup3。

基数估计算法的原理（续）

基数估计算法分类

- Linear 系（空间复杂度与基数上限呈线性关系，基数较小时表现好）
- LogLog 系（空间复杂度与基数上限呈线性关系，基数较大时表现好）
- 混合系（综合了以上两种派别，整体效果较好）

基数估计算法的原理（续）

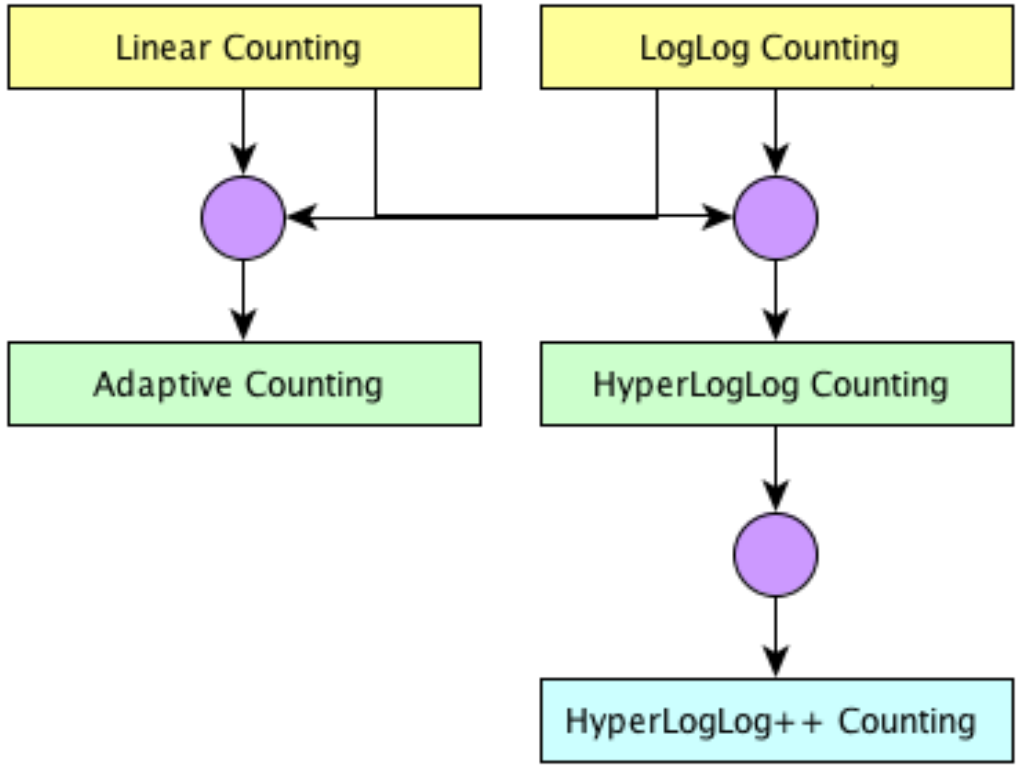


Figure: 基数估计算法家谱

基数估计算法的原理（续）

Linaer 系原理

- 以线性压缩的 bitmap 为数据结构
- 以空位置为统计量

- 特点
  - 基数较小时效果较好
  - 误差限一定时，空间复杂度为 $O(N_{max})$
  - 当 bitmap 满后会失效

基数估计算法的原理（续）

LogLog 系原理

- 以分桶 bitmap 为数据结构
- 以每桶元素二进制表示最长 0 前缀长度为统计量

- 特点
  - 基数较大时效果较好，基数小时误差很大
  - 误差限一定时，空间复杂度为 $O(log(log(N_{max})))$

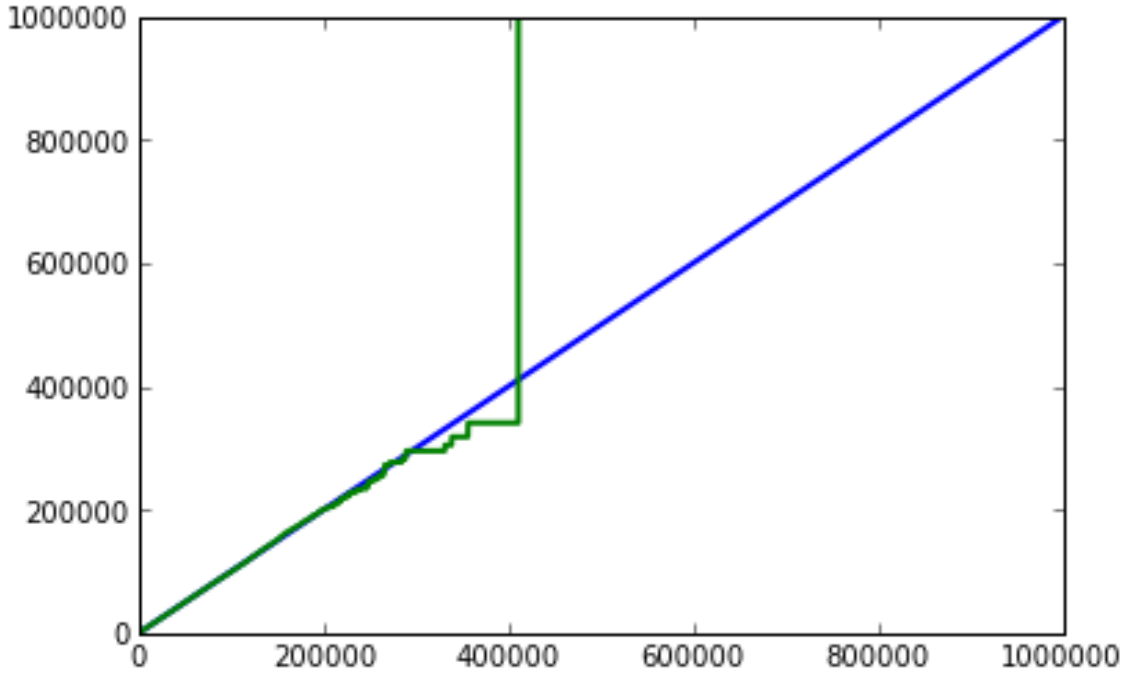
基数估计算法的原理（续）

混合系原理

- 综合了 Linear 系和 LogLog 系
- 对一些极端情况进行了工程性修正
- 特点
  - 总体效果较好，适合实际应用

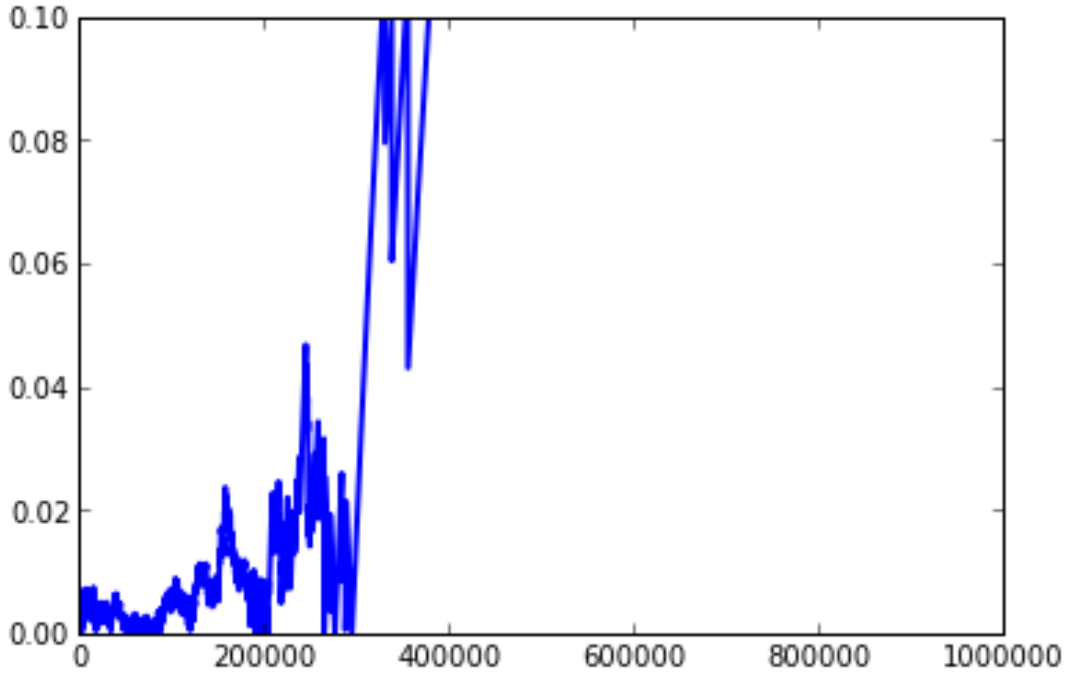
不同基数估计算法的实验效果

Linear Counting(p = 12)



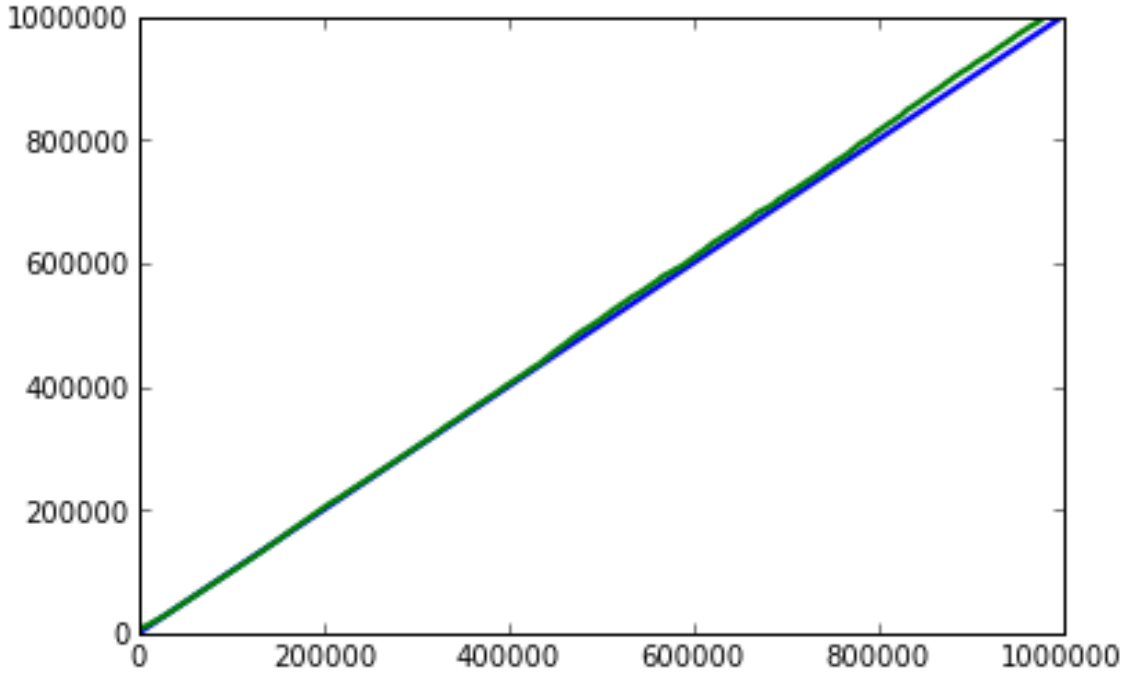
不同基数估计算法的实验效果(续)

Linear Counting(p = 12)



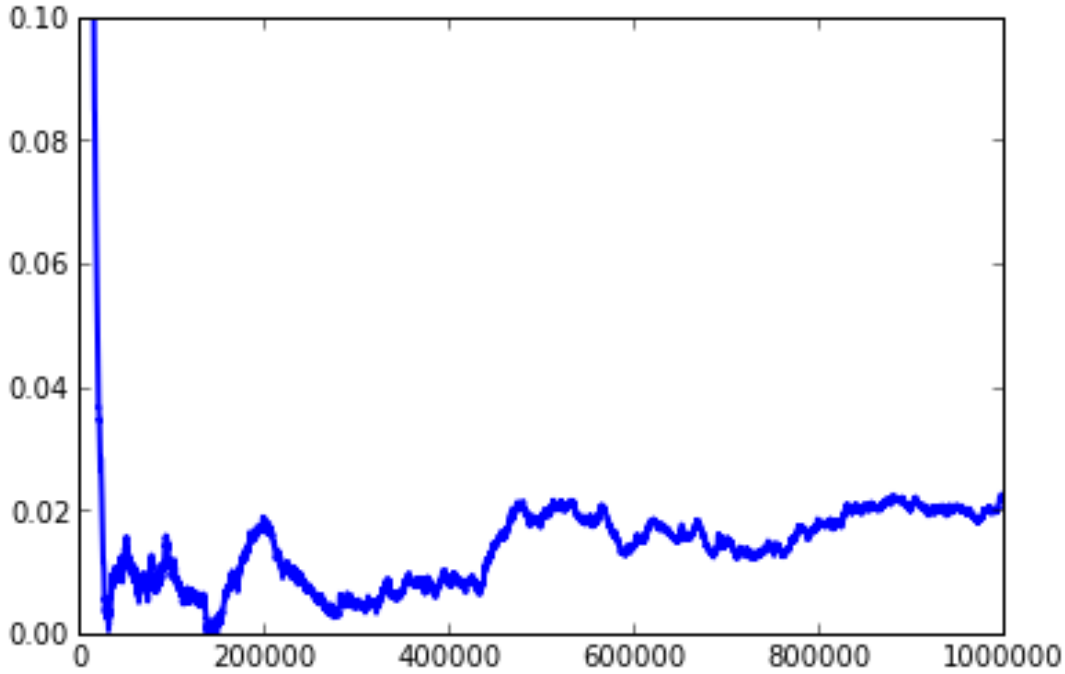
不同基数估计算法的实验效果(续)

LogLog Counting(p = 12)



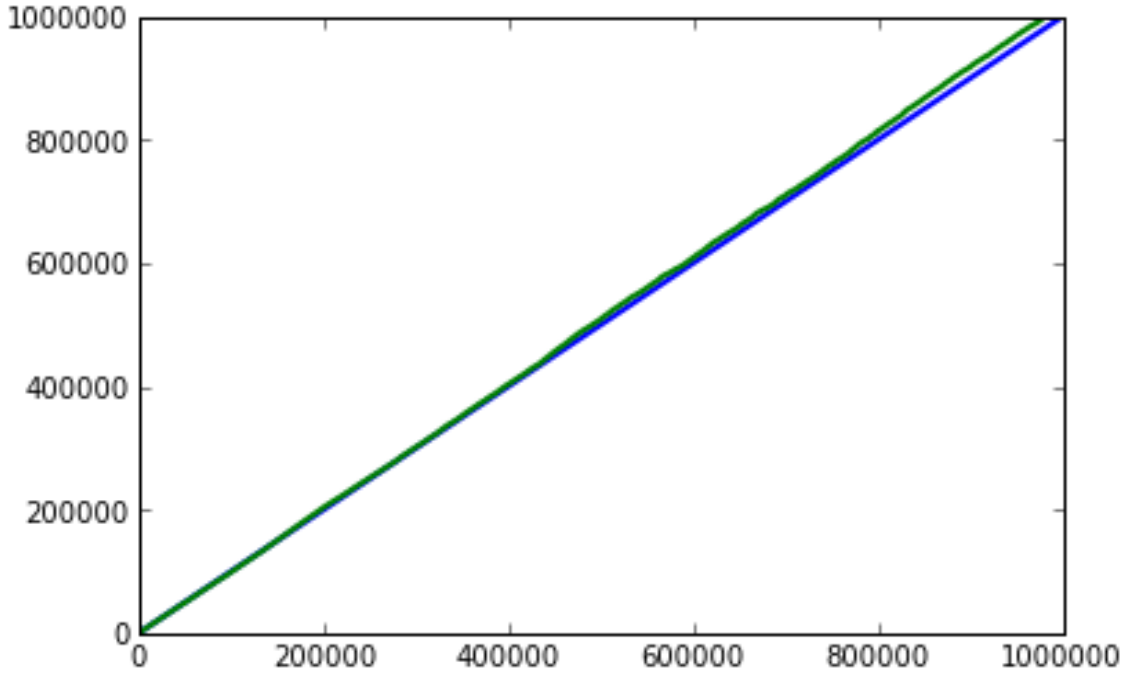
不同基数估计算法的实验效果(续)

LogLog Counting(p = 12)



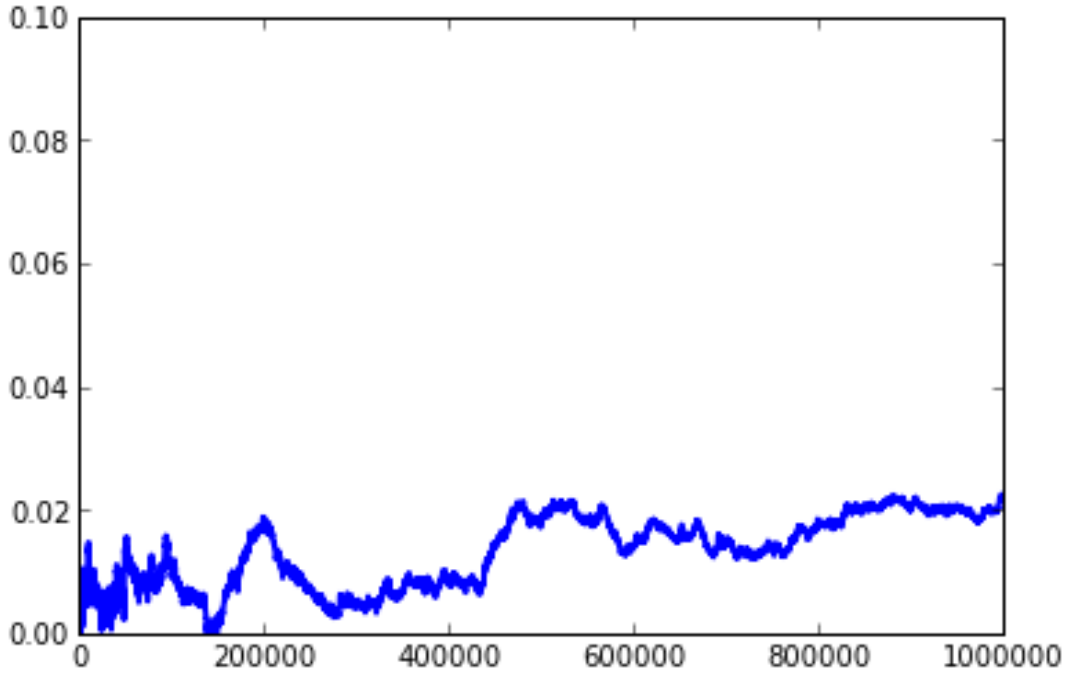
不同基数估计算法的实验效果(续)

Adaptive Counting(p = 12)



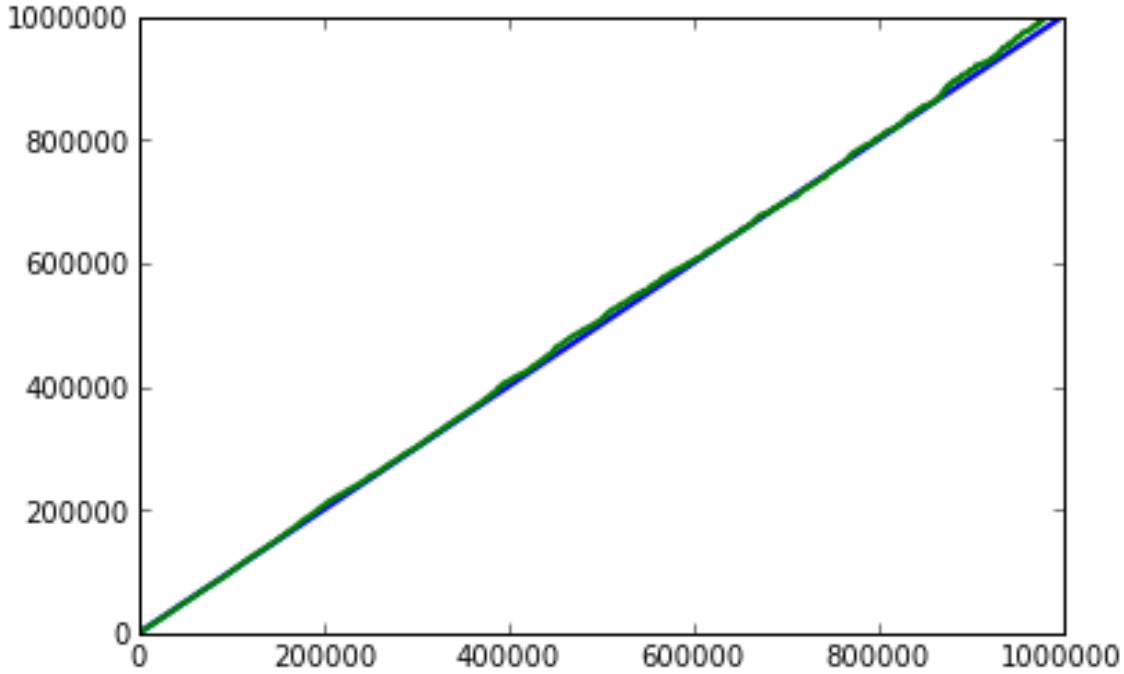
不同基数估计算法的实验效果(续)

Adaptive Counting(p = 12)



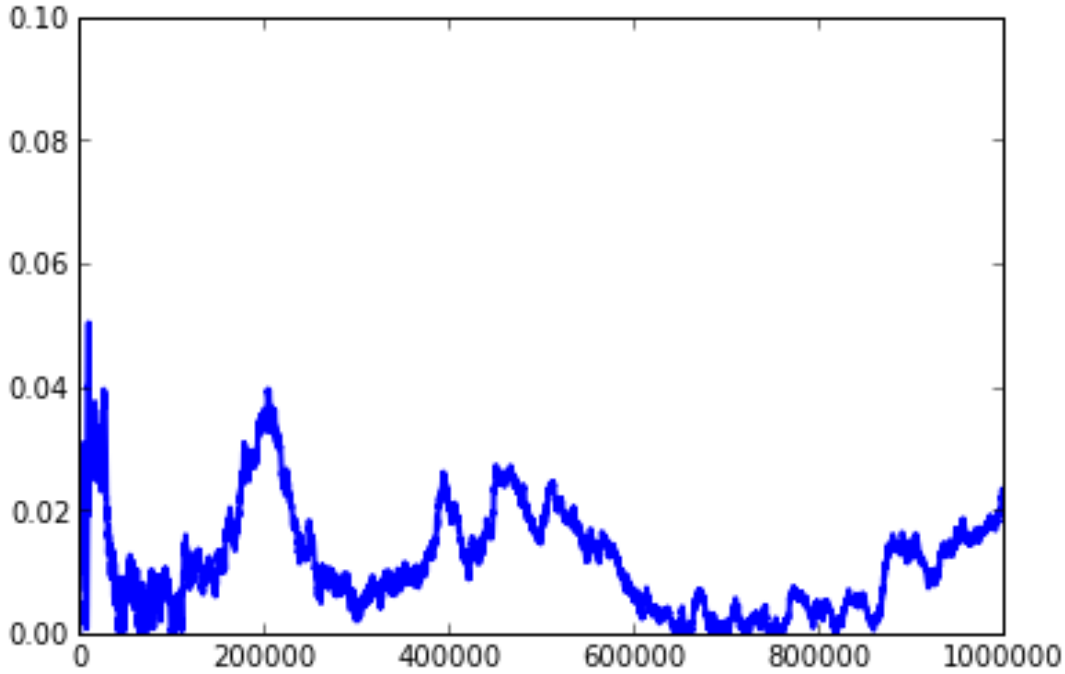
不同基数估计算法的实验效果(续)

HyperLogLog Counting(p = 12)



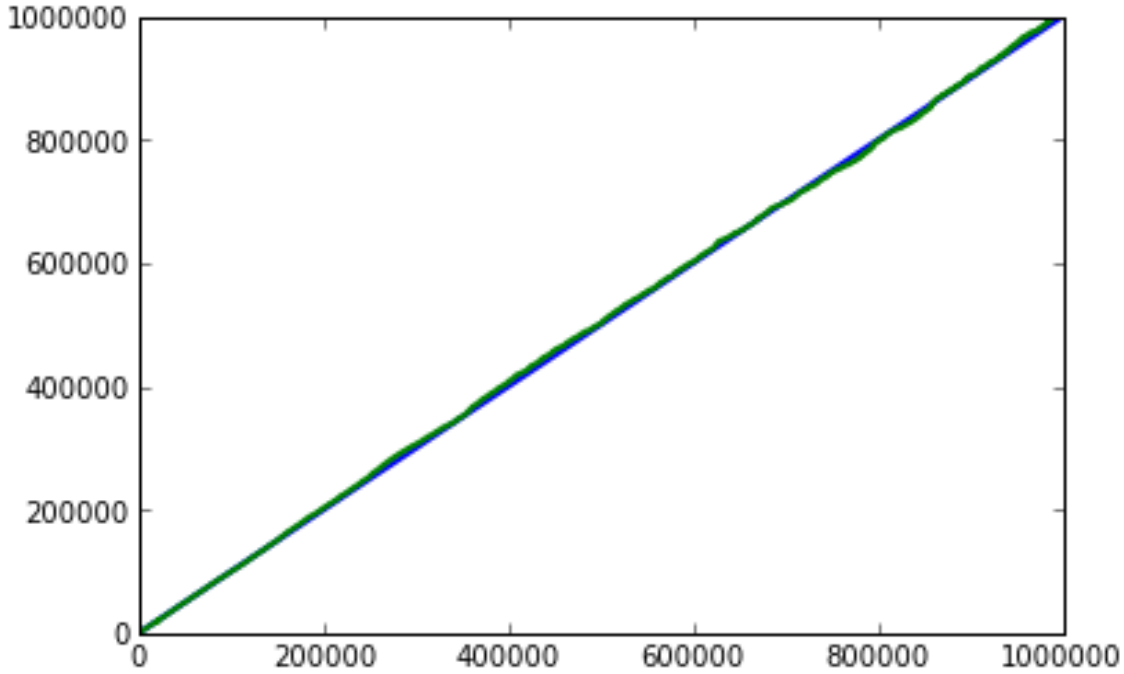
不同基数估计算法的实验效果(续)

HyperLogLog Counting(p = 12)



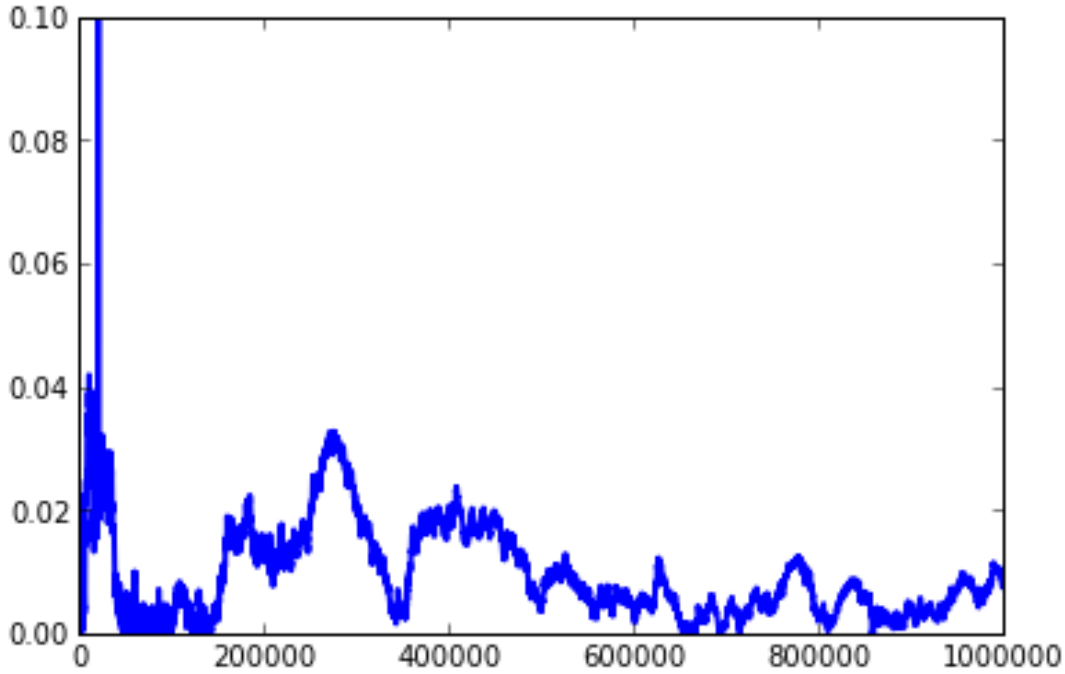
不同基数估计算法的实验效果(续)

HyperLogLog++ Counting(p = 12)



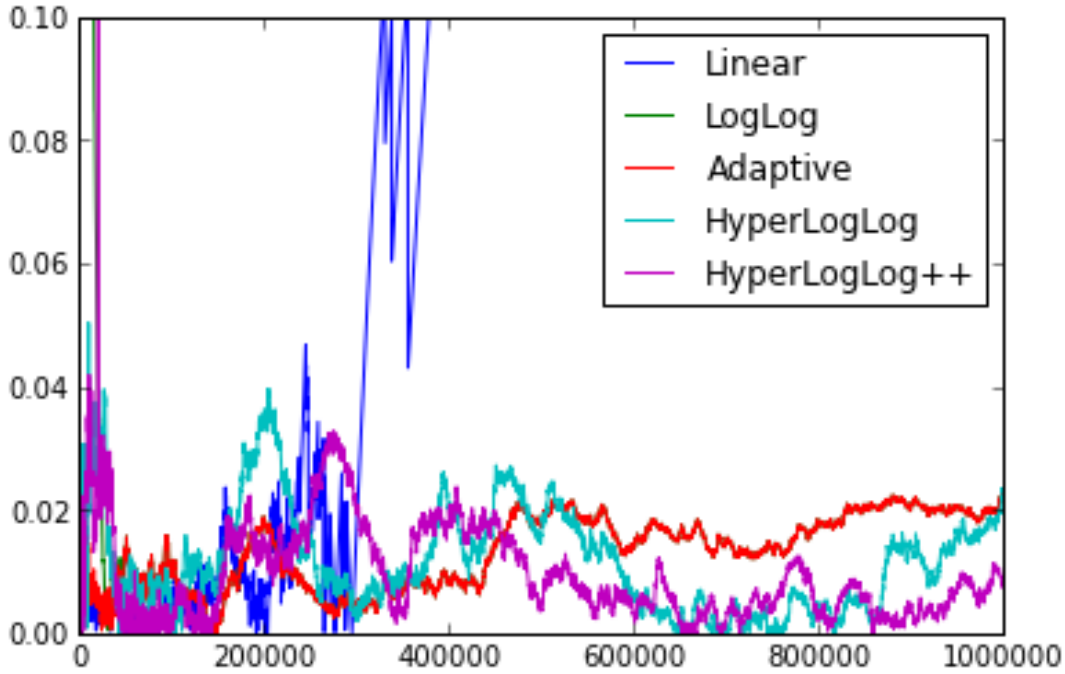
不同基数估计算法的实验效果(续)

HyperLogLog++ Counting(p = 12)



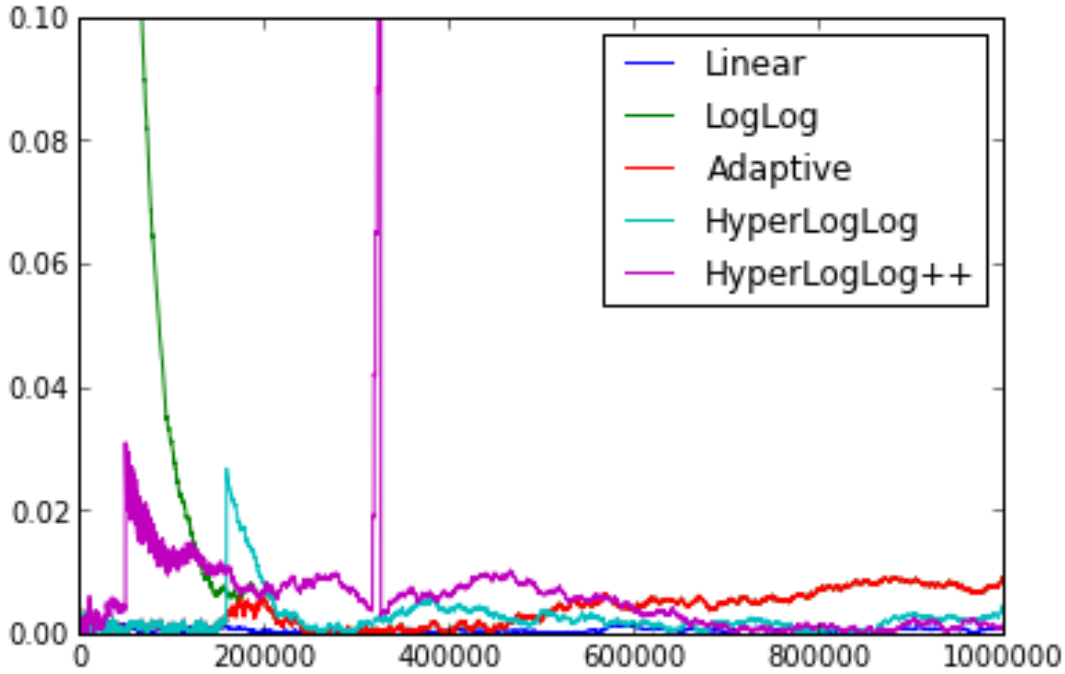
不同基数估计算法的实验效果（续）

误差叠加图（p=12）



不同基数估计算法的实验效果（续）

误差叠加图（p=16）



基数估计算法挑选的基本原则

- Linear Counting 和 LogLog Counting 由于分别在基数较大和基数较小（阈值可解析分析，具体方法和公式请参考后文列出的相关论文）时存在严重的失效，因此不适合在实际中单独使用。一种例外是，如果对节省存储空间要求不强烈，不要求空间复杂度为常数（Linear Counting 的空间复杂度为 O(n)，其它算法均为 O(1)），则在保证 bitmap 全满概率很小的条件下，Linear Counting 的效果要优于其它算法。

基数估计算法挑选的基本原则（续）

- 总体来看，不论哪种算法，提高分桶数都可以降低偏差和方差，因此总体来看基数估计算法中分桶数的选择是最重要的一个权衡——在精度和存储空间间的权衡。

基数估计算法挑选的基本原则（续）

- 实际中，Adaptive Counting 或 HyperLogLog Counting 都是不错的选择，前者偏差较小，后者对离群点容忍性更好，方差较小。

基数估计算法挑选的基本原则（续）

- Google 的 HyperLogLog Counting++ 算法属于实验性改进，缺乏严格的数学分析基础，通用性存疑，不宜在实际中贸然使用。

基数估计算法库

- Java: https://github.com/clearspring/stream-lib
- C/C++: https://github.com/chaoslawful/ccard-lib

相关论文

- K.-Y. Whang, B. T. Vander-Zanden, and H. M. Taylor. A Linear-Time Probabilistic Counting Algorithm for Database Applications. ACM Transactions on Database Systems, 15(2):208-229, 1990.
- Marianne Durand and Philippe Flajolet. LogLog counting of large cardinalities. In ESA03, volume 2832 of LNCS, pages 605-617, 2003.
- Min Cai, Jianping Pan, Yu K. Kwok, and Kai Hwang. Fast and accurate traffic matrix measurement using adaptive cardinality counting. In MineNet '05: Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data, pages 205-206, New York, NY, USA, 2005. ACM.

相关论文（续）

- P. Flajolet, E. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. Disc. Math. and Theor. Comp. Sci., AH:127-146, 2007.
- Stefan Heule, Marc Nunkesser, Alex Hall. HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm. In Proceedings of the EDBT 2013 Conference, ACM, Genoa, Italy.

网络资料

- http://blog.aggregateknowledge.com/
- http://blog.codinglabs.org/