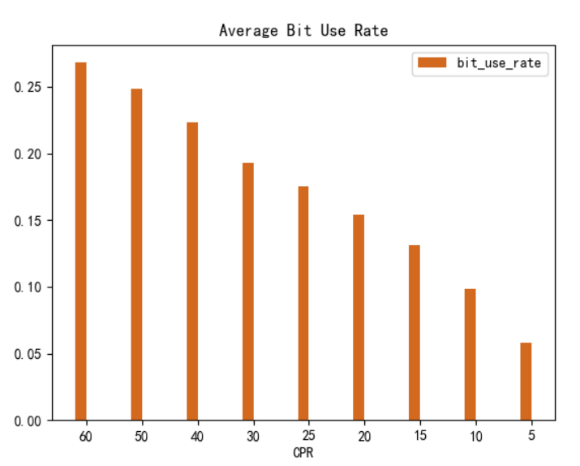
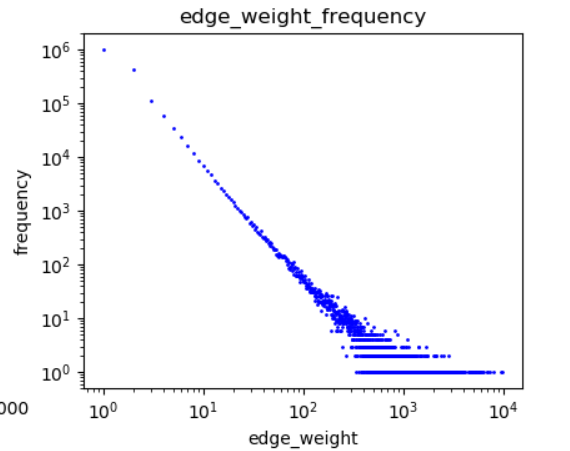
**说明文档**

**摘要：**

图流作为一种快速无限到达的边流序列，构成了一个随时间快速变化的图模型。由于其绝对规模的巨大和实时处理和响应多种查询语义的要求，使得摘要存储成为一种实际的图流存储模型。其中TCM和GSS（具体参见文件夹中的前两篇论文）是两种经典的图流摘要存储模型。但是，现有的存储结构都是通过对图流规模的预估然后使用固定大小的计数counter，由于图流无界性的特点，实现权值可动态扩展的结构成为现实的要求。针对这样一个问题，我们设计了一个对于图流权值高效可扩展的框架——Yurt sketch，并将其在TCM和GSS上分布做了实现。

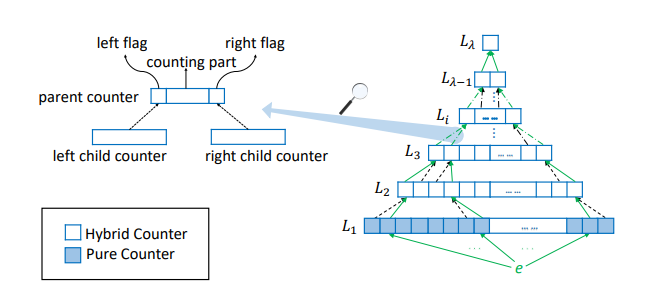
**基本思路：**

由于图流数据中的边的权值分布具有倾斜性（大部分的边的权值较小，而小部分的边的权值很大，图一a是使用真实数据集得到的实验结果），这使得统一分配相同的counter大小会造成比特位的大量浪费。争对这样的一种情况，Pyramid Sketch（参见文件夹中的第三篇论文），提出了一种共享高位溢出的结构，使得在相同的内存大小的情况下，得到了更精确的结果，图二展示了这个结构的基本思想



1. (b)

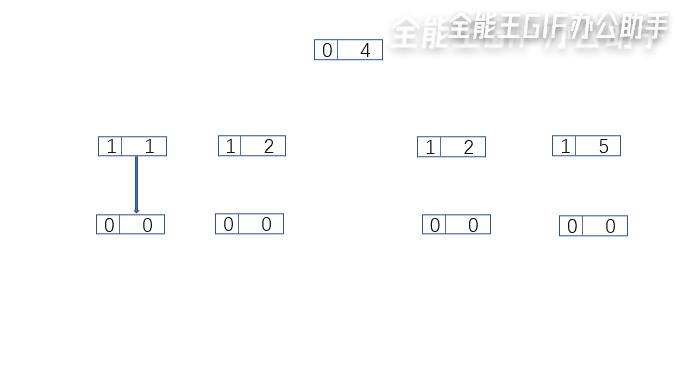
图一



图二

但是Pyramid sketch并没有实现对于权值的可扩展，在此基础上，我们提出一种新型的支持高效权值可扩展的图流摘要存储结构——Yurt。我们可以看出随着权值的不断增大，金字塔结构的counter溢出情况越来越糟糕，由于溢出冲突带来的查询误差（当下层对应的counter都溢出时会共享高位的counter）越来越大，最终将变成一个误差不可接受的结构。

争对这样的一种情况，我们通过监控金字塔底层counter的溢出情况，当counter溢出导致的溢出冲突使得存储结构的查询误差不可用时，我们通过将上层counter的权值下放到相应的下层counter中来缓解溢出冲突，如图三所示。



图三

**在TCM上的实现：**

我们将这样的一种框架在TCM上进行了实现，并做了一些优化。

* 优化一：

图流作为一个二维的图模型，在实际的应用中，往往存在节点的度的倾斜分布的情况，在TCM结构中，高度点的边都会映射到矩阵的同一行或者同一列中，这使得，权值较高的counter在哈希矩阵中呈现出行列分布的聚集性。如图四所示



图四

对于这样的一种情况，为了尽量的避免溢出冲突的发生，在设计金字塔层与层之间的对应规则时，我们采用一种分区扫描的对应规则如图五所示。（其中相同颜色代表相对应的counter）这样，对应同一个上层counter的不同下层counter都来自不同的行和列。



图五

* 优化二

另外，我们为了实现对于权值扩展的更加细粒度的控制，在下放时，我们增加的金字塔底座的counter的大小设置为1个bit，相当于每次都对相应的counter做一个右移操作。实例如图六所示。



图六

**在GSS上的实现：**

另外，我们将pyramid结构在GSS上做了实现，考虑到GSS是一个精确的查询结构，我们加入了如下的两种优化机制来尽可能地避免溢出冲突。

**优化一：备选地址重选策略**

当新插入地边即将发生溢出冲突时，我们将发生溢出冲突地另外一条边踢出，这一过程持续直到没有溢出冲突的发生



**优化一：上层counter共享策略**

多个不同的下层counter不使用原来固定的对应规则，而是使用共享的counter对应规则。在多个共享的counter中选取一个还未接受到溢出的counter进行溢出。



**源代码：**

文件夹sourceHeadfiles包含了TCM，GSS，和Yurt在其上的实现。