Abstract

1.问题：视觉处理成本很高->资源管理至关重要->VideoStorm提出

2.VideoStorm两大Characteristic(two properties of video analytics queries relevant to resource management.):

①resource-quality tradeoff with multi-dimensional configurations

多维度配置下资源使用率和质量之间的权衡

②variety in quality and lag goals.

不同查询请求在质量要求和延迟忍受的差异

3. offline profiler：generates query resource-quality profile

online scheduler allocates resources to queries to maximize performance on quality

and lag, in contrast to the commonly used fair sharing of resources in clusters.

为了解决诸如查询空间大，质量、耗资权衡棘手，集群内资源分配计算复杂等问题，将系统工作过程分为在线和离线两阶段：离线阶段负责在避免全局搜索的条件下为用户请求找出其对应的profile，即少数帕累托最优解配置，再交给在线阶段在此范围内周期性地为请求更改配置，调整资源分配。

Introduction

1. Key to achieving the potential of these cameras is effectively analyzing the live video streams.（目标）

2. **knobs**: Vision algorithms typically contain various parameters. E.g. video resolution, frame rate, and internal algorithmic parameters, such as the size of the sliding window to search for objects in object detectors.（基本概念）

3. **query configuration**: A combination of the knob values. The configuration space grows exponentially with the number of knobs.（基本概念）

4. Resource-quality trade-off with multi-dimensional configurations: Resource demand can be reduced by changing configurations (e.g., changing the resolution and sliding window size) but they typically also lower the output quality.

（权衡问题）

5. Resource demand can be reduced by changing configurations (e.g., changing the resolution and sliding window size) but they typically also lower the output quality.

（不同query对accuracy的要求和lag的容忍是不同的）

6. Core of VideoStorm: VideoStorm contains a scheduler that efficiently generates the query’s resource-quality profile for its different knob configurations, and then jointly maximizes the quality and minimizes the lag of streaming video queries. In doing so, it uses the generated profiles, and lag and quality goals. It allocates resources to each query and picks its configuration (knob values) based on the allocation.

（与一般的视频分析系统不同点在于，其集群资源**是经过调度的，不是公平的**）

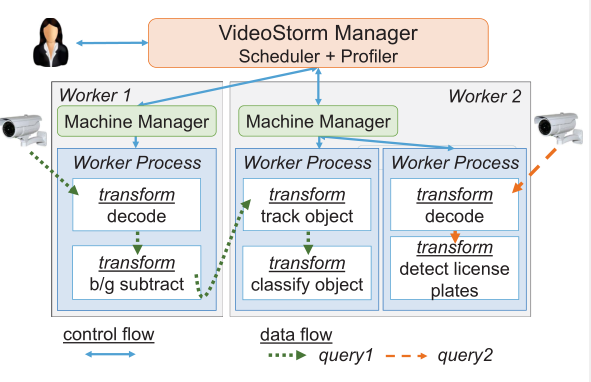
7. In the **offline** phase, we use an efficient profiler to get the resource-quality profile of queries without exploring the entire combinatorial space of configurations. Using **greedy search and domain-specific sampling**, we identify a handful of knob configurations on the **Pareto boundary** of the profile. The scheduler in the **online** phase, thus, has to consider only these configurations.

（工作分为在线、离线两个阶段，离线阶段负责通过profiler，以贪心搜索和特定域采样避免在指数级空间内搜索最优配置，最终得到了配置解的帕累托边界（边界上都是帕累托最优解），再让在线阶段只负责考虑其中这些帕累托最优解配置即可。）

8.在线阶段用到了模型预测控制MPC来预测将来的延迟并送给效用函数（包括延迟、质量）

二.System Description

问题：什么是transform?处理流的一个过程？



1.VideoStorm集群由一个Manager和多个Worker机器构成。

Query:可以看作是不断流向集群的由transform作为节点的有向无环图

Machine Manager：每个Worker都有一个，负责启动工作进程以及向主Manager汇报资源使用情况和工作状态

VideoStorm Manager:根据Machine Manager反馈的信息决定资源的分配（给并发的多个query）

2.Query规范

Query是由pipeline和transform构成的串（json?）

问题：什么是pipeline?一个由transform构成的流水线？



一个transform由id,class\_name,parameter(knob)构成；

Input\_transform\_id指示流的来源，即上一级transform。最初的transform不用该字段，而是直接与源相连（由ip和port绑定）；

可变的knob前面要有@

Class\_name表示实现该transform的类名？

三、Making the Case for Resource Allocation

1.问题：

“We considered streaming databases with approximation [19,37,68] as a starting point for our solution. However, they only consider the sampling rate of data streams and used established analytical models [38] to calculate the quality and resource demand.”

我们将流式数据库的近似值[19,37,68]作为我们解决方案的起点。但它们只考虑数据流的采样率，并使用已建立的分析模型[38]来计算质量和资源需求。相比之下，视觉查询是更复杂的黑盒，有更多的旋钮，并且没有已知的分析模型。此外，它们每次只优化一个查询，而我们的重点是调度多个并发查询。

“In contrast to approximate SQL query processing,there are no analytical models to estimate the relationship between resource demand and quality of video queries and it depends on the specific video feeds.”

与近似的SQL查询处理相比，没有分析模型来估计资源需求和视频查询质量之间的关系，它取决于特定的视频提要。例如，降低视频分辨率可能不会降低OpenALPR的质量，如果相机被放大的足够大。因此，需要使用具有代表性的视频样本来分析查询。

为啥这玩意能跟数据库和SQL扯上关系？什么是流式数据库？

查询网上的信息：

传统的数据处理系统通常是对已经存储在数据库系统或文件系统等其它存储系统中的完整静态数据集进行计算和分析，显然不适合这类持续生成的、无限的、动态的数据流。而且传统的批处理技术在数据生成到数据被处理之间通常会有比较长的时间间隔，然而在如今激烈竞争、复杂多变的商业环境下， 营销时机转瞬即逝，风险防控必须分秒必争，商业决策要求快速精准，因此数据的处理必须在更短的时间内得到结果，最好是能够做到实时处理。

在这种背景下，实时流处理技术开始在越来越多的场景下逐渐替代批处理技术，并在现代的数据分析技术栈中占据核心的位置。 流处理能够自然地对连续的数据流进行建模，与对静态数据的查询和分析不同，它能够随着新数据的到来实时地对计算结果进行更新，这使得「数据生成」-> 「获取数据洞察」-> 「采取行动」之间没有延迟， 从而让企业在激烈的市场竞争中始终保持主动和领先优势。

我们认为， **不同于其它数据库系统将静态的数据集（表或文档等）作为基本的存储和处理单元，流数据库是以动态的连续数据流作为基本对象，以实时性作为主要特征的数据库。流数据库是数据库在流时代的重新架构和设计**。

2.The desirable properties of a scheduler for video analytics are: (1) allocate more resources to queries whose qualities will improve more, (2) allow queries with built-up lag in their processing to “catch up,” and (3) adjust query configuration based on the resource allocated.

resource-quality profile: a small number of configurations on the Pareto curve of the profile

视频分析的调优程序的理想特性是:(1)分配更多的资源给质量提高更多的查询，(2)允许在处理过程中有构建延迟的查询“赶上”，(3)根据分配的资源调整查询配置。

3.challenges: First, the configuration space of a query can be large and there are no analytical models to estimate the resource demand and result quality of each configuration.

Second, trading off between the lag and quality goals of queries is tricky, making it challenging to define scheduling objectives across all queries in the cluster. Third, resource allocation across all queries in the cluster each with many configurations is computationally

intractable, presenting scalability challenges.

系统设计的困难：首先，查询的配置空间可能很大，而且没有分析模型来估计每个配置的资源需求和结果质量。其次，在查询的延迟和质量目标之间进行权衡是一件棘手的事情，这使得在集群中的所有查询中定义调度目标具有挑战性。第三，集群中所有具有许多配置的查询的资源分配在计算上是难以处理的，这带来了可伸缩性的挑战。

五、Resource-Quality Profile Estimation

1.The query profiler has two goals. 1) Select a small subset of configurations (Pareto boundary) from the resource-quality space, and 2) Compute the query profile, Pk, i.e., the resource demand and result quality of the selected configurations. The profile is computed either against a labeled dataset or using the initial parts of the video relative to a “golden” query configuration which might be expensive but is known to produce high-quality results.

2.offline的**greedy local search和hill climbing**:

以X(c) = Q(c) − βD(c)表示对一个configuration的质量和耗资间的权衡（β为权衡系数）。对某一请求，随机选取k种配置计算其X值，并从中选取X值最大的配置作为起始配置c（这样可以防止从一个成本高昂的配置开始，尽可能减小搜索的资源耗费）。之后每次改变其中的一个knob的一个值（所谓的临近配置n）并与原配置的X值比较，如果n的X值更大，则称n优于c，并将n作为c的新值，重复以上步骤，直到附近找不到更优的配置，即找到了局部最优解。之后再另选取新的起始配置c，继续重复以上步骤，即确定了其他局部最优解。问题：这些局部最优解加起来就构成了当前请求配置的帕累托最优解集？，并传递给在线阶段。

问题：得到的局部最优解的总和是否就是帕累托最优解集？即X(n)>X(c)是否等价于Q(n)>=Q(c)&&D(n)<=D(c)？如果不是，作者是如何从中提取出帕累托最优解的？

六、Resource Management

1.Utility Function**效用方程：**结合质量（quality）与延迟（lag）

When deployed as a “service” in the public cloud, utility will represent the revenue the cluster operator generates by executing the query; penalties and bonuses in utility translate to loss

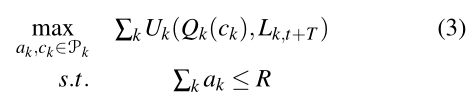
and increase in revenue. Therefore, maximizing the sum of utilities maximizes revenue.

√问题：如何理解效用方程？

基准效用与高质量奖励与高延迟惩罚之间的加权和？衡量一个query的综合表现的指标？

2.用单个请求的效用表示整个集群的整体效用：如何使整个集群的表现效用达到最大？

可以通过使最小单个效用最大或使单个效用之和最大来实现，由此产生了两个**最优问题**：



在总资源限制范围内，使得总效益最大。这也是一个**背包问题。**这样以来就有两种思路：DP和贪心。

这里作者采取贪心算法：首先计算uk/ak，即各个请求的单位收益，再从高单位收益到低单位收益依次选择请求，直到用完资源或者所有请求都给与需求的资源量。**同时通过从0开始逐步逼近总资源量，以规避不可行结果。**

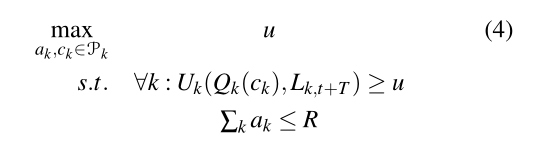
问题：这不就是真解吗？为啥作者说这是greedy approximation？还有，为什么逼近R可以规避不可行结果？什么情况下会出现不可行结果？

自己尝试回答问题一：因为这虽然是个背包问题，具有最优子结构和贪心选择性质，但其还具有实际应用背景，属于背包问题的变种：即为并发的请求分配资源。真解是只要保证在不超过资源限制的前提下效用和最大即可，但我们不能简单地选择该真解，因为我们的资源调度虽然是不公平的，但依然需要对所有请求做出回应，而真解可能会导致一部分单位收益很小的请求始终得不到任何资源而被挂起，这显然是不应该的。所以作者采取了这样的近似处理：每一步迭代中，为每个请求都分配按一极小步长递增的资源，manager控制各个请求随即根据资源选择合适的配置，进而产生一定效益。选择单位效益最大的请求。

产生新的问题：Our heuristic starts with ak = 0 and in each step we consider increasing ai (for all queries i) by a small ∆ (say, 1% of a core) 意思是对当前所有请求都给予相同分量的资源？

and consider all configurations of ci ∈ Pi. Among these options, we select query i (and corresponding ci) with largest increase in utility.“选出”是什么意思？将query i从请求中取出，下一步不再考虑i？单个请求的单位效益不会随着得到的资源量而变化吗？

近似解的思路大致是这样的吗：为每个请求都分配一定的资源，但优先选择满足单位效益高的请求的最优配置。



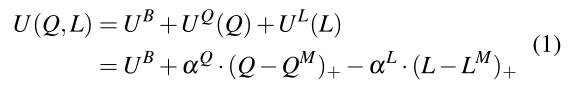
问题：这两个优化问题是同时满足？是的话不会矛盾？还是只选择其中一种？

3.资源分配：**模型预测控制(MPC)**

由于当资源分配量少于请求所需量时会产生延迟，故系统存在延迟堆叠效应，不能控制当前的配置，而只能谋求控制未来一小段时间的配置，因而需要对未来的延迟进行预测，而作者采用了MPC的方法进行预测。



再将该预测延时代入效应方程，即可表示预测效应（收益）：



4.查询请求的放置与迁移

分布式集群中需要对请求进行合适的放置，并在适当时候进行服务迁移。

the suitability of placing a query q on machine m by computing a score for each of the following goals: high utilization, load balancing, and spreading low-lag queries.

问题：什么叫queries的packing?高利用率与负载均衡是否冲突？

High utilization in the cluster can be achieved by packing queries in to machines, thereby minimizing fragmentation and wastage of resources.

Spreading load across the cluster ensures that each machine has spare capacity to handle changes in demand. We therefore prefer to place q on a machine m with the smallest utilization.

迁移发生的条件：The final score sq,m is the average of the three scores. For each new query q, we place it on a machine with the largest sq,m. For each existing query q, we migrate from machine m0 to a new machine m1 only if its score improves substantially; i.e., s(q,m1) − s(q,m0) > τ.

5.改进

①考虑资源需求预测的不准确性（如需求量随视频内容有较大变化时）并纳入滞后预测因子

②大多数请求只需在一台机器即可胜任，故将短期内资源重分配的步骤转交给各个工作机器，机器内的请求构成新的请求集，机器的总资源量代替原来集群的总资源量，在机器尺度上为请求进行资源分配，而原来的主调度器只负责调度请求的放置与迁移。

问题：是这样理解的吗？

③包含DAG的多转换查询需要跨机器放置，此时查询的放置还要考虑预期的跨机器数据流和网络链接容量。

七、implementation

①Flow control:跨查询转换的流控制，最小化查询管道内的缓冲，防止解码帧过大而内存溢出以及使帧迅速响应配置的变化。

②Migration：首先在目标机器上运行待迁移查询的副本并复制输入流，较短时间后再停止原机器上的查询。

③Resource Enforcement：采用类似linux容器的技术对资源量强行限制。（cgroup？）

④transform类需要实现的接口：

byte[] Process(header, data) 用于处理输入byte数组data（帧）并返回处理结果（也是byte数组）

Update(key, value) 用于运行时更改knob的值以改变查询的配置。