FALL UPDATE
PDL PACKET

NEWSLETTER ON PDL ACTIVITIES AND EVENTS • FALL 2018
http://www.pdl.cmu.edu/

PDL CONSORTIUM MEMBERS

Alibaba Group
Amazon
Datrium
Dell EMC
Facebook
Google
Hewlett Packard Enterprise
Hitachi, Ltd.
IBM Research
Intel Corporation
Micron
Microsoft Research
MongoDB
NetApp, Inc.
Oracle Corporation
Salesforce
Samsung Information Systems America
Seagate Technology
Two Sigma
Veritas
Western Digital

CONTENTS

THE PDL PACKET

EDITOR
Joan Digney
CONTACTS
Greg Ganger
PDL Director
Bill Courtright
PDL Executive Director
Karen Lindenfelser
PDL Administrative Manager
The Parallel Data Laboratory
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3891
TEL 412-268-6716
FAX 412-268-3010

http://www.pdl.cmu.edu/Publications/

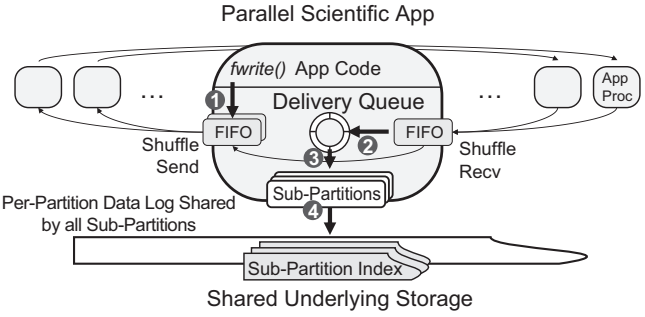
SELECTED RECENT PUBLICATIONS

Scaling Embedded In-Situ Indexing with DeltaFS

Qing Zheng, Charles D. Cranor, Danhao Guo, Gregory R. Ganger, George Amvrosiadis, Garth A. Gibson, Bradley W. Settlemyer, Gary Grider & Fan Guo

SC18, November 11-16, 2018, Dallas, Texas, USA.

Analysis of large-scale simulation output is a core element of scientific inquiry, but analysis queries may experience significant I/O overhead when the data is not structured for efficient retrieval. While in-situ processing allows for improved time-to-insight for many applications, scaling in-situ frameworks to hundreds of thousands of cores can be difficult in practice. The DeltaFS in-situ indexing is a new approach for in-situ processing of massive amounts of data to achieve efficient point and small-range queries. This paper describes the challenges and lessons learned when scaling this in-situ processing function to hundreds of thousands of cores. We propose techniques for scalable all-to-all communication that is memory and bandwidth efficient, concurrent indexing, and specialized LSM-Tree formats. Combining these techniques allows DeltaFS to control the cost of in-situ processing while maintaining 3 orders of magnitude query speedup when scaling alongside the popular VPIC particle-in-cell code to 131,072 cores.



Updated DeltaFS in-situ indexing pipeline design with a new delivery queue structure, and a multi-way indexing mechanism.

Stratus: Cost-aware Container Scheduling in the Public Cloud

Andrew Chung, Jun Woo Park & Gregory R. Ganger

ACM Symposium on Cloud Computing, 2018 (SoCC'18), Carlsbad, CA October 11-13, 2018.

Stratus is a new cluster scheduler specialized for orchestrating batch job execution on virtual clusters, dynamically allocated collections of virtual machine instances on public IaaS platforms. Unlike schedulers for conventional clusters, Stratus focuses primarily on dollar cost considerations, since public clouds provide effectively unlimited, highly heterogeneous resources allocated on demand. But, since

continued on page 2

continued from page 1

resources are charged-for while allocated, Stratus aggressively packs tasks onto machines, guided by job runtime estimates, trying to make allocated resources be either mostly full (highly utilized) or empty (so they can be released to save money). Simulation experiments based on cluster workload traces from Google and Two Sigma show that Stratus reduces cost by 17–44% compared to state-of-the-art approaches to virtual cluster scheduling.

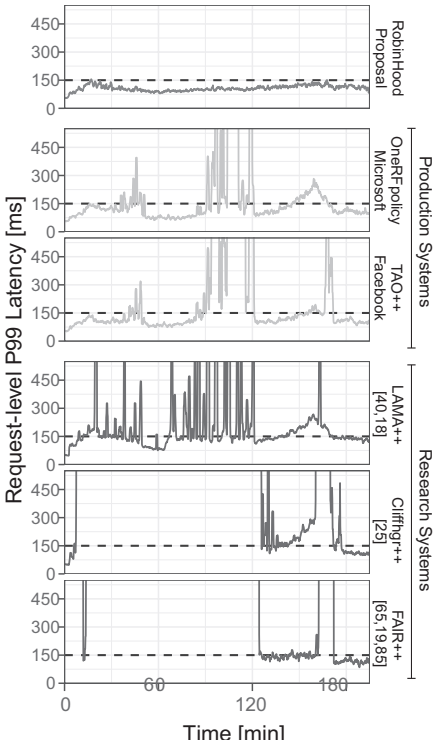
RobinHood: Tail Latency Aware Caching—Dynamic Reallocation from Cache-Rich to Cache-Poor

Daniel S. Berger, Benjamin Berg, Timothy Zhu, Siddhartha Sen & Mor Harchol-Balter

13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18). October 8–10, 2018, Carlsbad, CA, USA.

Tail latency is of great importance in user-facing web services. However, maintaining low tail latency is challenging, because a single request to a web application server results in multiple queries to complex, diverse backend services (databases, recommender systems, ad systems, etc.). A request is not complete until all of its queries have completed. We analyze a Microsoft production system and find that backend query latencies vary by more than two orders of magnitude across backends and over time, resulting in high request tail latencies.

We propose a novel solution for maintaining low request tail latency: repurpose existing caches to mitigate the effects of backend latency variability, rather than just caching popular data. Our solution, RobinHood, dynamically reallocates cache resources from the cache-rich (backends which don't affect request tail latency) to the cache-poor (backends which affect request tail latency). We evaluate RobinHood with production traces on a 50- server cluster with 20 different backend systems. Surprisingly, we find that RobinHood can directly address tail latency even if working sets are much larger than the cache size. In the presence of load spikes, RobinHood meets a 150ms P99 goal 99.7% of the time, whereas the next best policy meets this goal only 70% of the time.



Comparison of the P99 request latency of RobinHood, two production caching systems, and three stateof-the-art research caching systems, which we emulated in our testbed. All systems are subjected to three load spikes. We draw a dashed line at 150ms, which is the worst latency under RobinHood.

The Parallel Persistent Memory Model

Guy E. Blelloch, Phillip B. Gibbons, Yan Gu, Charles McGuffey & Julian Shun

SPAA '18, July 16–18, 2018, Vienna, Austria.

We consider a parallel computational model, the Parallel Persistent Memory model, comprised of P processors, each with a fast local ephemeral memory of limited size, and sharing a large persistent memory. The model allows for each processor to fault at any time (with bounded probability), and possibly restart. When a processor faults, all of its state and local ephemeral memory is lost, but the persistent memory remains. This model is motivated by upcoming non-volatile memories that are nearly as fast as existing random access memory, are accessible at the granularity of cache lines, and have the capability of surviving power outages. It is further motivated by the observation that in large parallel systems, failure of processors and their caches is not unusual.

We present several results for the model, using an approach that breaks a computation into capsules, each of which can be safely run multiple times. For the single-processor version we describe how to simulate any program in the RAM, the external memory model, or the ideal cache model with an expected constant factor overhead. For the multiprocessor version we describe how to efficiently implement a work-stealing scheduler within the model such that it handles both soft faults, with a processor restarting, and hard faults, with a processor permanently failing. For any multithreaded fork-join computation that is race free, write-after-read conflict free and has W work, D depth, and C maximum capsule work in the absence of faults, the scheduler guarantees a time bound on the model of $O(W/P_A + DP/P_A \lceil \log_{1/(Cf)} W \rceil)$ in expectation, where P is the maximum number of processors, P_A is the average number, and $f \leq 1/(2C)$ is the probability a processor faults between successive persistent memory accesses. Within the model, and using the proposed methods, we develop efficient algorithms for parallel prefix sums, merging, sorting, and matrix multiply.

continued on page 3

October 2018
Best Student Paper at SoCC '18!

Congratulations to Andrew Chung and Jun Woo Park, who submitted the Best Student Paper to SoCC '18. Their paper at the Symposium for Cloud Computing, titled "Stratus: Cost-aware Container Scheduling in the Public Cloud," discusses cost considerations of a new cluster scheduler specialized to orchestrate batch job execution on virtual clusters, which dynamically allocates collections of virtual machine instances on public IaaS platforms.

September 2018
PDL Alum Wei Dai Winner of Pittsburgh Business Times 30 under 30 Award!

Wei (David) Dai, who graduated with his Ph.D. in Machine Learning from CMU in 2018 has been listed as one of Pittsburgh's 30 under 30 by the Pittsburgh Business Times. Wei is now the Senior Director of Engineering at Petuum, where they build scalable machine learning platforms for enterprises to easily create and manage complex ML workflows.

-- photo credit, Joe Wojcik

July 2018
PDL Team Tests New File System on Trinity Supercomputer

A team from the Carnegie Mellon Parallel Data Lab (PDL) recently completed work with Los Alamos National Lab simulating physical phenomena involving as many as a trillion individual particles. Their project used the Trinity supercomputer to test a new file system that created a trillion files in just two minutes, allowing them to retrieve data one to five thousand times faster than conventional methods. The team included George Amvrosiadis, Chuck Cranor, Greg Ganger, and Ph.D. student Qing Zheng; the PDL's Garth Gibson; and Los Alamos National Lab's Brad Settlemyer and Gary Grider. Read the full article in Wired here.

-- ECE News July 31, 2018

May 2018
Best Paper at SIGMOD 2018!

The Carnegie Mellon Database Group is pleased to announce that their latest paper "SuRF: Practical Range Query Filtering with Fast Succinct Tries" has won 2018 SIGMOD Best Paper Award. The paper's lead author was CMU CSD Ph.D. Huanchen Zhang. This work was in collaboration with CMU professors Dave Andersen and Andy Pavlo, CMU post-doc Hyeontaek Lim, TUM visiting scholar Viktor Leis, Hewlett Packard Labs' Distinguished Technologist Kimberly Keeton, and Intel Labs' senior research scientist Michael Kaminsky.

RECENT PUBLICATIONS

Putting the "Micro" Back in Microservice

Sol Boucher, Anuj Kalia, and David G. Andersen & Michael Kaminsky

2018 USENIX Annual Technical Conference (USENIX ATC '18). July 11–13, 2018, Boston, MA.

Modern cloud computing environments strive to provide users with fine-grained scheduling and accounting, as well as seamless scalability. The most recent face to this trend is the "serverless" model, in which individual functions, or microservices, are executed on demand. Popular implementations of this model, however, operate at a relatively coarse granularity, occupying resources for minutes at a time and requiring hundreds of milliseconds for a cold launch. In this paper, we describe a novel design for providing "functions as a service" (FaaS) that attempts to be truly micro: cold launch times in microseconds that enable even finer-grained resource accounting and support latency-critical applications. Our proposal is to eschew much of the traditional serverless infrastructure in favor of language-based isolation. The result is microsecond-granularity launch latency, and microsecond-scale preemptive scheduling using high-precision timers.

continued on page 7

DISSERTATION ABSTRACT: The Design and Implementation of a Non-Volatile Memory Database Management System

Joy James Prabhu Arulraj
Carnegie Mellon University,, SCS

PhD Defense — July 13, 2018

This dissertation explores the implications of emergent non-volatile memory (NVM) technologies for database management systems (DBMSs). The advent of NVM will fundamentally change the dichotomy between volatile memory and durable storage in DBMSs. These new NVM devices are almost as fast as DRAM, but all writes to it are potentially persistent even after power loss. Existing DBMSs are unable to take full advantage of this technology because their internal architectures are predicated on the assumption that memory is volatile. With NVM, many of the components of legacy DBMSs are unnecessary and will degrade the performance of the data intensive applications.

We present the design and implementation of a new DBMS tailored specifically for NVM. The dissertation focuses on three aspects of a DBMS: (1) logging and recovery, (2) storage management, and (3) indexing. Our primary contribution in this dissertation is the design of a new logging and recovery protocol, called write-behind logging, that improves the availability of the system by more than two orders of magnitude compared to the ubiquitous write-ahead logging protocol. Besides improving availability, we demonstrate that write-behind logging extends the lifetime and increases the space utilization of the NVM device. Second, we propose a new storage engine architecture that leverages the durability and byte-addressability properties of NVM to avoid unnecessary data duplication. Third, the dissertation presents the design of a range index tailored for NVM that supports near-instantaneous recovery without requiring special-purpose recovery code.

Larry Rudolph of Two Sigma talks about cloud computing at the PDL Consortium day of seminars, May 8, 2018.

DISSERTATION ABSTRACT: Practical Concurrency Testing or: How I Learned to Stop Worrying and Love the Exponential Explosion

Ben Blum
Carnegie Mellon University, SCS

PhD Defense — October 17, 2018

Concurrent programming presents a challenge to students and experts alike because of the complexity of multi-threaded interactions and the difficulty to reproduce and reason about bugs. Stateless model checking is a concurrency testing approach which forces a program to interleave its threads in many different ways, checking for bugs each time. This technique is powerful, in principle capable of finding any nondeterministic bug in finite time, but suffers from exponential explosion as program size increases. Checking an exponential number of thread interleavings is not a practical or predictable approach for programmers to find concurrency bugs before their project deadlines.

In this thesis, I develop several new techniques to make stateless model checking more practical for human use. I have built Landslide, a stateless model checker specializing in undergraduate operating systems class projects. Landslide extends the traditional model checking algorithm with a new framework for automatically managing multiple state spaces according to their estimated completion times, which I show quickly finds bugs should they exist and also quickly verifies correctness otherwise. I evaluate Landslide's suitability for inexpert use by presenting the results of many semesters providing it to students in 15-410, CMU's Operating System Design and Implementation class, and more recently, students in similar classes at the University of Chicago and Penn State University. Finally, I extend Landslide with a new concurrency model for hardware transactional memory, and evaluate several real-world transactional benchmarks to show that stateless model checking can keep up with the developing concurrency demands of real-world programs.

DISSERTATION ABSTRACT: Framework Design for Improving Computational Efficiency and Programming Productivity for Distributed Machine Learning

Jin Kyu Kim
Carnegie Mellon University, SCS

PhD Defense — September 26, 2018

Machine learning (ML) methods are used to analyze data in a wide range of areas, such as finance, e-commerce, medicine, science, and engineering, and the size of machine learning problems has grown very rapidly in terms of data size and model size in the era of big data. This trend drives industry and academic communities toward distributed machine learning that scales out ML training in a distributed system for completion in a reasonable amount of time. There are two challenges in implementing distributed machine learning: computational efficiency and programming productivity. The traditional data-parallel approach often leads to suboptimal training performance in distributed ML due to data dependencies among model parameter updates and nonuniform convergence rates of model parameters. From the perspective of an ML programmer, distributed ML programming requires substantial

continued on page 5

continued from page 4

development overhead even with high-level frameworks because they require an ML programmer to switch to a different mental model for programming from a familiar sequential programming model.

The goal of my thesis is to improve the computational efficiency and programming productivity of distributed machine learning. In an efficiency study, I explore model update scheduling schemes that consider data dependencies and nonuniform convergence speeds of model parameters to maximize convergence per iteration and present a runtime system STRADS that efficiently execute model update scheduled ML applications in a distributed system. In a productivity study, I present familiar sequential-like programming API that simplifies conversion of a sequential ML program into a distributed program without requiring an ML programmer to switch to a different mental for programming and implement a new runtime system STRADS-Automatic Parallelization(AP) that efficiently executes ML applications written in our API in a distributed system.

THESIS PROPOSAL: Efficient and Programmable Distributed Shared Memory Systems for Machine Learning Training

Jinliang Wei, SCS
October 5, 2018

Machine learning training involves frequent and often sparse updates to a large number of numerical values called model parameters. Many distributed training systems have resorted to using distributed shared memory (DSM) (e.g. Parameter Server) for efficient sparse access and in-place updates. Compared to traditional programs, machine learning applications tolerate bounded error, which presents opportunities for trading off learning progress for higher computation throughput. In this thesis, I develop efficient and programmable distributed learning systems, by exploiting this trade-off in the design of distributed shared memory systems, along with parallelization and static and dynamic scheduling.

Thanks to this tolerance to bounded error, a machine learning program can often be parallelized without strictly preserving data dependence. Parallel workers may thus observe inconsistent model parameter values compared to a serial execution. More frequent communication to propagate updates and fresher parameter values may reduce such inconsistency, while incurring higher inter-machine communication overhead. I present a communication management mechanism that automates communication using spare network bandwidth and prioritizes messages according to their importance in order to reduce error due to inconsistency while retaining high computation throughput.

When each model update reads and writes to only a subset of model parameters, it is possible to achieve an efficient parallelization while preserving critical data dependence, exploiting sparse parameter access. Existing systems require substantial programmer effort to take advantage of this opportunity. In order to achieve dependence-preserving parallelization without imposing a huge burden on application programmers, I present a system Orion that provides parallel for-loops on distributed shared memory and parallelizes loops with loop-carried dependence.

At last, I propose to explore dynamic scheduling for dynamic control flow in dataflow systems such as TensorFlow. In TensorFlow, the computation graph is statically partitioned and assigned with computation devices. Static device placement is suboptimal as the operators' load can no longer be determined statically due to dynamic control flow. A suboptimal static device placement may result in imbalanced load and extra communication. It is promising to address the deficiency of static device placement by dynamically scheduling operations based on their load at runtime.

Andy Pavlo and Geert Bosch (MongoDB) discuss database research at the 2018 PDL Retreat. Photo credit, Nicolas Viennot.

THESIS PROPOSAL: Low-Latency, Low-Cost Machine Learning Systems on Large-Scale, Highly-Distributed Data

Kevin Hsieh, ECE
August 9, 2018

The explosive advancement of machine learning (ML) has been the engine of many important applications. The success of an ML-driven application depends on two key factors: low latency and low cost. However, achieving low-latency and low-cost ML is particularly challenging when the ML processes depend on real-world, large-scale data (e.g., user activities, pictures, and videos), which are massive and highly distributed.

In this thesis proposal, we identifiy three major challenges to achieve low-latency and low-cost ML on massive and highly-distributed data. We describe three research directions that address these challenges with system-level solutions. Our solutions improve the latency and cost of ML on masive and highly-distributed data by one to two orders of magnitude.

First, many ML systems leverage state-of-the-art deep neural networks (DNNs) to process large and continuosly growing data (e.g., videos, audios, pictures) with the goal to answer "after the fact" queries such as: identify video frames with objects of certain classes (cars, bags). However, supporting such queries incurs high cost at ingest time or high latency at

continued on page 6

continued from page 5

query time. We present Focus, a system providing both low-cost and low-latency queries over large datasets, using video queries as the case study.

Second, when ML data are highly distributed (e.g., distributed in many data centers across the world), massive communication overhead can drastically slow down an ML system and introduce substantial cost. To this end, we introduce a new, general geo-distributed ML training system, Gaia, that enables efficient communication between data centers by dynamically eliminating insignificant communication while still guaranteeing the correctness of ML algorithms.

THESIS PROPOSAL: Rethinking Cross-layer Abstractions to Enhance Programmability, Portability, and Performance

Nandita Vijaykumar, ECE
August 9th, 2018

The last decades have seen tremendous change and growth across all levels of the computing stack—applications, programming models, compilers, runtime systems, and the hardware architecture. These changes are driven by recent trends, including the push towards domain-specific specialization in hardware and software, consolidation of multiple applications on the same platform via system virtualization, and a new era of data-intensive computation. Programmability, performance portability, and resource efficiency have emerged as critical challenges in harnessing complex and diverse architectures today to obtain high performance and energy efficiency. While there is abundant research, and thus significant improvements, at different levels of the stack that address these very challenges, the interfaces/abstractions between the levels of the computing stack have largely remained the same.

This thesis makes a case for rethinking the cross-layer abstractions in the new landscape of fast-evolving hardware and software. While today the cross-layer abstractions are primarily designed for program functionality and correctness, we explore how richer interfaces can make a significant difference in how we optimize for programmability, performance portability, and resource efficiency across the computing stack. We propose 4 different approaches to designing richer abstractions between the application, system software, and hardware architecture: (i) Expressive Memory: A unifying cross-layer abstraction to express and communicate higher-level program semantics from the application to the underlying system/architecture to enhance memory optimization; (ii) The Locality Descriptor: A cross-layer abstraction to express and exploit data locality in GPUs; (iii) Zorua: A framework to decouple the programming model from management of on-chip resources and parallelism in GPUs; (iv) Assist Warps: A helper-thread abstraction to dynamically leverage underutilized compute/memory bandwidth in GPUs to perform useful work. We describe each concept and propose the research questions to be addressed in this thesis.

THESIS PROPOSAL: Distribution-based cluster scheduling

Jun Woo Park, SCS
May 16, 2018

This thesis seeks to propose and evaluate a scheduler that can leverage full distributions (e.g.,the histogram of observed runtimes or resource usage) rather than single point estimates. Knowing point estimates, such as how long each job will execute, enables a scheduler to more effectively pack jobs with diverse time concerns (e.g., deadline vs. the-sooner-the-better) and placement preferences on heterogeneous cluster resources. But, existing schedulers use single-point estimates (e.g., mean or median of a relevant subset of historical runtimes), and we show that they are fragile in the face of real-world estimate error profiles. In particular, analysis of job traces from three different large-scale cluster environments shows that, while the runtimes of many jobs can be predicted well, even state-of-the-art predictors have wide error profiles with 8-23% of predictions off by a factor of two or more. Instead of reducing relevant history to a single point, a distribution provides much more information (e.g., variance, possible multi-modal behaviors, etc.) and allows the scheduler to make more robust decisions. By considering the range of possible runtimes and resource usage for a job, and their likelihoods, the scheduler can explicitly consider various potential outcomes from each possible scheduling option and select an option based on optimizing the expected outcome.

Greg Ganger and Micheal Abd-El-Malek (PDL alumnus, now with Google) catch up at the 2018 PDL Retreat. Photo credit, Nicolas Viennot.

THESIS PROPOSAL: Improving ML Applications in Shared Computing Environments

Aaron Harlap, ECE
May 16, 2018

Statistical machine learning (ML) has become a powerful building block for modern services, scientific endeavors and enterprise processes. We focus on the major subset of ML approaches that employ iterative algorithms to determine model parameters that best fit a given set of input data. Such algorithms iterate over the input data, refining their current best estimate of the parameter values to converge on a final solution.

The expensive computations required for training ML applications often makes it desirable to run them in a distributed manner in shared computing environments (e.g Amazon EC2, Microsoft Azure, in-house shared clusters). Distributed training of ML applications commonly require the resources involved to maintain parameter

continued on page 7

continued from page 6

data (solution state), evenly distribute work, synchronize progress and communicate amongst each other in order for the ML application to function effectively.

Shared computing environments introduce a number of challenges including uncorrelated performance jitter, heterogeneous resources, transient resources and limited bandwidth. In our work we focus on improving the efficiency, reducing cost and reducing runtime of training ML applications in shared computing environments by addressing the challenges described.

THESIS PROPOSAL: Towards Space-Efficient High-Performance In-Memory Search Structures

Huanchen Zhang,CS
April 30, 2018

This thesis seeks to address the challenge of building space-efficient yet high- performance in-memory search structures, including indexes and filters, to allow more efficient use of memory in OLTP databases. We show that we can achieve this goal by first designing fast static structures that leverage succinct data structures to approach the information-theoretic optimum in space, and then using the "hybrid index" architecture to obtain dynamicity with bounded and modest cost in space and performance.

To obtain space-efficient yet high-performance static data structures, we first in- troduce the Dynamic-to-Static rules that present a systematic way to convert existing dynamic structures to smaller immutable versions. We then present the Fast Succinct Trie (FST) and its application, the Succinct Range Filter (SuRF), to show how to leverage theories on succinct data structures to build static search structures that consume space close to the information-theoretic minimum while performing comparably to uncompressed indexes. To support dynamic operations such as inserts, deletes, and updates, we introduce the dual-stage hybrid index architecture that preserves the space efficiency brought by a compressed static index, while amortizing its performance overhead on dynamic operations by applying modifications in batches.

In the proposed work, we seek opportunities to further shrink the size of in-memory indexes by co-designing the indexes with the in-memory tuple storage. We also propose to complete the hybrid index work by extending the techniques to support concurrent indexes.

continued from page 3

Mainstream: Dynamic Stem-Sharing for Multi-Tenant Video Processing

Angela H. Jiang, Daniel L.K. Wong, Christopher Canel, Lilia Tang, Ishan Misra, Michael Kaminsky, Michael A. Kozuch, Padmanabhan Pillai, David G. Andersen & Gregory R. Ganger

2018 USENIX Annual Technical Conference (USENIX ATC '18). July 11–13, 2018, Boston, MA.

Mainstream is a new video analysis system that jointly adapts concurrent applications sharing fixed edge resources to maximize aggregate result quality. Mainstream exploits partial-DNN (deep neural network) compute sharing among applications trained through transfer learning from a common base DNN model, decreasing aggregate per-frame compute time. Based on the available resources and mix of applications running on an edge node, Mainstream automatically determines at deployment time the right trade-off between using more specialized DNNs to improve per-frame accuracy, and keeping more of the unspecialized base model to increase sharing and process more frames per second. Experiments with several datasets and event detection tasks on an edge node confirm that Mainstream improves mean event detection F1-scores by up to 47% relative to a static approach of retraining only the last DNN layer and sharing all others ("Max-Sharing") and by 87X relative to the common approach of using fully independent per-application DNNs ("No-Sharing").

Geriatrix: Aging What You See and What You Don't See — A File System Aging Approach For Modern Storage Systems

Saurabh Kadekodi, Vaishnavh Nagarajan, Gregory R. Ganger & Garth A. Gibson

2018 USENIX Annual Technical Conference (USENIX ATC '18). July 11–13, 2018, Boston, MA.

File system performance on modern primary storage devices (Flash-based SSDs) is greatly affected by aging of the free space, much more so than were mechanical disk drives. We introduce Geriatrix, a simple-to-use profile driven file system aging tool that induces target levels of fragmentation in both allocated files (what you see) and remaining free space (what you don't see), unlike previous approaches that focus on just the former. This paper describes and evaluates the effectiveness of Geriatrix, showing that it recreates both fragmentation effects better than previous approaches. Using Geriatrix, we show that measurements presented in many recent file systems papers are higher than should be expected, by up to 30% on mechanical (HDD) and up to 80% on Flash (SSD) disks. Worse, in some cases, the performance rank ordering of file system designs being compared are different from the published results.

continued on page 8

continued from page 7

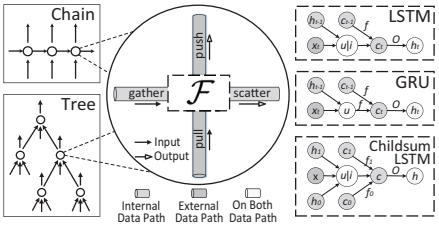
Geriatrix will be released as open source software with eight built-in aging profiles, in the hopes that it can address the need created by the increased performance impact of file system aging in modern SSD-based storage.

Cavs: An Efficient Runtime System for Dynamic Neural Networks

Shizhen Xu, Hao Zhang, Graham Neubig, Wei Dai, Jin Kyu Kim, Zhijie Deng, Qirong Ho, Guangwen Yang & Eric P. Xing

2018 USENIX Annual Technical Conference (USENIX ATC '18). July 11–13, 2018, Boston, MA.

Recent deep learning (DL) models are moving more and more to dynamic neural network (NN) architectures, where the NN structure changes for every data sample. However, existing DL programming models are inefficient in handling dynamic network architectures because of: (1) substantial overhead caused by repeating dataflow graph construction and processing every example; (2) difficulties in batched execution of multiple samples; (3) inability to incorporate graph optimization techniques such as those used in static graphs. In this paper, we present "Cavs", a runtime system that overcomes these bottlenecks and achieves efficient training and inference of dynamic NNs. Cavs represents a dynamic NN as a static vertex function F and a dynamic instance-specific graph G. It avoids the overhead of repeated graph construction by only declaring and constructing F once, and allows for the use of static graph optimization techniques on pre-defined operations in F. Cavs performs training and inference by scheduling the execution of F following the dependencies in G, hence naturally exposing batched execution opportunities over different samples. Experiments comparing Cavs to state-of-the-art frameworks for dynamic NNs (TensorFlow Fold, PyTorch and



Cavs represents a dynamic structure as a dynamic input graph G (left) and a static vertex function F (right).

DyNet) demonstrate the efficacy of our approach: Cavs achieves a near one order of magnitude speedup on training of dynamic NN architectures, and ablations verify the effectiveness of our proposed design and optimizations.

Litz: Elastic Framework for High-Performance Distributed Machine Learning

Aurick Qiao, Abutalib Aghayev, Weiren Yu, Haoyang Chen, Qirong Ho, Garth A. Gibson & Eric P. Xing

2018 USENIX Annual Technical Conference (USENIX ATC '18). July 11–13, 2018, Boston, MA.

Machine Learning (ML) is an increasingly popular application in the cloud and data-center, inspiring new algorithmic and systems techniques that leverage unique properties of ML applications to improve their distributed performance by orders of magnitude. However, applications built using these techniques tend to be static, unable to elastically adapt to the changing resource availability that is characteristic of multi-tenant environments. Existing distributed frameworks are either inelastic, or offer programming models which are incompatible with the techniques employed by high-performance ML applications.

Motivated by these trends, we present Litz, an elastic framework supporting distributed ML applications. We categorize the wide variety of techniques employed by these applications into three general themes — stateful workers, model scheduling, and relaxed consistency — which are collectively supported by Litz's programming model. Our implementation of Litz's execution system transparently enables elasticity and low-overhead execution. We implement several popular ML applications using Litz, and show that they can scale in and out quickly to adapt to changing resource availability, as well as how a scheduler can leverage elasticity for faster job completion and more efficient resource allocation. Lastly, we show that Litz enables elasticity without compromising performance, achieving competitive performance with state-of-the-art non-elastic ML frameworks.

A Case for Packing and Indexing in Cloud File Systems

Saurabh Kadekodi, Bin Fan, Adit Madan, Garth A. Gibson & Gregory R. Ganger

10th USENIX Workshop on Hot Topics in Cloud Computing, July 9, 2018, Boston, MA.

Small (kilobyte-sized) objects are the bane of highly scalable cloud object stores. Larger (at least megabyte-sized) objects not only improve performance, but also result in orders of magnitude lower cost, due to the current operation-based pricing model of commodity cloud object stores. For example, in Amazon S3's current pricing scheme, uploading 1GiB data by issuing 4KiB PUT requests (at 0.0005 cents each) is approximately 57X more expensive than storing that same 1GiB for a month. To address this problem, we propose client-side packing of small immutable files into gigabyte-sized blobs with embedded indices to identify each file's location. Experiments with a packing implementation in Alluxio (an open-source distributed file system) illustrate the potential ben-

continued on page 9

continued from page 8

efits, such as simultaneously increasing file creation throughput by up to 60X and decreasing cost to 1/25000 of the original.

Tributary: Spot-dancing for Elastic Services with Latency SLOs

Aaron Harlap, Andrew Chung, Alexey Tumanov, Gregory R. Ganger & Phillip B. Gibbons

2018 USENIX Annual Technical Conference. July 11–13, 2018 Boston, MA, USA.

The Tributary elastic control system embraces the uncertain nature of transient cloud resources, such as AWS spot instances, to manage elastic services with latency SLOs more robustly and more cost-effectively. Such resources are available at lower cost, but with the proviso that they can be preempted en masse, making them risky to rely upon for business-critical services. Tributary creates models of preemption likelihood and exploits the partial independence among different resource offerings, selecting collections of resource allocations that satisfy SLO requirements and adjusting them over time, as client workloads change. Although Tributary's collections are often larger than required in the absence of preemptions, they are cheaper because of both lower spot costs and partial refunds for preempted resources. At the same time, the often-larger sets allow unexpected workload bursts to be absorbed without SLO violation. Over a range of web service workloads, we find that Tributary reduces cost for achieving a given SLO by 81–86% compared to traditional scaling on non-preemptible resources, and by 47–62% compared to the high-risk approach of the same scaling with spot resources.

On the Diversity of Cluster Workloads and its Impact on Research Results

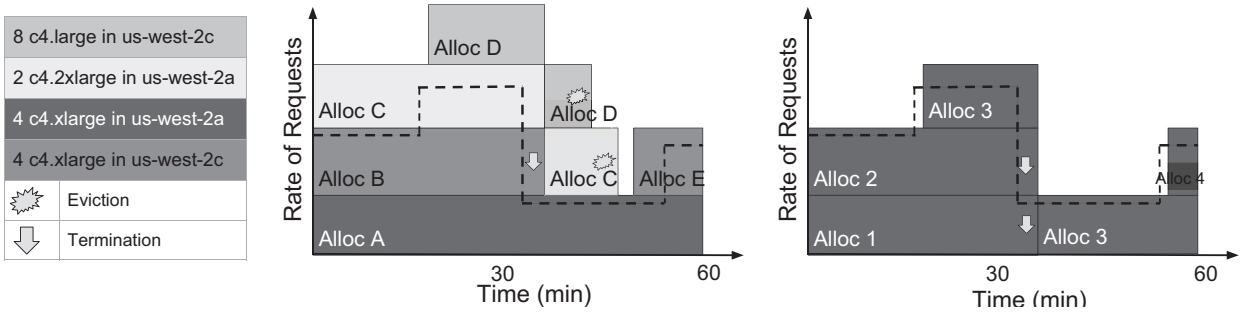
George Amvrosiadis, Jun Woo Park, Gregory R. Ganger, Garth A. Gibson, Elisabeth Baseman & Nathan DeBardeleben

2018 USENIX Annual Technical Conference (ATC '18), Boston, MA, July 11-13, 2018.

Six years ago, Google released an invaluable set of scheduler logs which has already been used in more than 450 publications. We find that the scarcity of other data sources, however, is leading researchers to overfit their work to Google's dataset characteristics. We demonstrate this overfitting by introducing four new traces from two private and two High Performance Computing (HPC) clusters. Our analysis shows that the private cluster workloads, consisting of data analytics jobs expected to be more closely related to the Google workload, display more similarity to the HPC cluster workloads. This observation suggests that additional traces should be considered when evaluating the generality of new research.

To aid the community in moving forward, we release the four analyzed traces, including: the longest publicly available trace spanning all 61 months of an HPC cluster's lifetime and a trace from a 300,000-core HPC cluster, the largest cluster with a publicly available trace. We present an analysis of the private and HPC cluster traces that spans job characteristics, workload heterogeneity, resource utilization, and failure rates. We contrast our findings with the Google trace characteristics and identify affected work in the literature. Finally, we demonstrate the

continued on page 10



Figures (b) and (c) show how Tributary and AutoScale handle a sample workload respectively. Figure (a) is the legend for (b) and (c), color-coding each allocation. The black dotted lines in (b) and (c) signify the request rates over time. At minute 15, the request rate unexpectedly spikes and AutoScale experiences "slow" requests until completing integration of additional resources with 3. Tributary, meanwhile, had extra resources meant to address preemption risk in C, providing a natural buffer of resources that is able to avoid "slow" requests during the spike. At minute 35, when the request rate decreases, Tributary terminates B, since it believes that B has the lowest probability of getting the free partial hour. It does not terminate D since it has a high probability of eviction and is likely to be free; it also does not terminate C since it needs to maintain resources. AutoScale, on the other hand, terminates both 2 and 3, incurring partial cost. At minute 52, the request rate increases and Tributary again benefits from the extra buffer while AutoScale misses its latency SLO. In this example, Tributary has less "slow" requests and achieves lower cost than AutoScale because AutoScale pays for 3 and for the partial hour for both 1 and 2 while Tributary only pays for A and the partial hour for B since C and D were preempted and incur no cost.

continued from page 9

importance of dataset plurality and diversity by evaluating the performance of a job runtime predictor using all four of our traces and the Google trace.

Mosaic: Enabling Application-Transparent Support for Multiple Page Sizes in Throughput Processors

Rachata Ausavarungnirun, Joshua Landgraf, Vance Miller, Saugata Ghose, Jayneel Gandhi, Christopher J. Rossbach & Onur Mutlu

ACM SIGOPS Operating System Review - Special Topics, Vol. 52, Issue 1, July 2018

Contemporary discrete GPUs support rich memory management features such as virtual memory and demand paging. These features simplify GPU programming by providing a virtual address space abstraction similar to CPUs and eliminating manual memory management, but they introduce high performance overheads during (1) address translation and (2) page faults. A GPU relies on high degrees of thread-level parallelism (TLP) to hide memory latency. Address translation can undermine TLP, as a single miss in the translation lookaside buffer (TLB) invokes an expensive serialized page table walk that often stalls multiple threads. Demand paging can also undermine TLP, as multiple threads often stall while they wait for an expensive data transfer over the system I/O (e.g., PCIe) bus when the GPU demands a page. In modern GPUs, we face a trade-off on how the page size used for memory management affects address translation and demand paging. The address translation overhead is lower when we employ a larger page size (e.g., 2MB large pages, compared with conventional 4KB base pages), which increases TLB coverage and thus reduces TLB misses. Conversely, the demand paging overhead is lower when we employ a smaller page size, which decreases the system I/O bus transfer latency. Support for multiple page sizes can help relax the page size trade-off so that address translation and demand paging optimizations work together synergistically. However, existing page coalescing (i.e., merging base pages into a large page) and splintering (i.e., splitting a large page into base pages) policies require costly base page migrations that undermine the benefits multiple page sizes provide. In this paper, we observe that GPGPU applications present an opportunity to support multiple page sizes without costly data migration, as the applications perform most of their memory allocation en masse (i.e., they allocate a large number of base pages at once). We show that this en masse allocation allows us to create intelligent memory allocation policies which ensure that base pages that are contiguous in virtual memory are allocated to contiguous physical memory pages. As a result, coalescing and splintering operations no longer need to migrate base pages.

We introduce Mosaic, a GPU memory manager that provides application-transparent support for multiple page sizes. Mosaic uses base pages to transfer data over the system I/O bus, and allocates physical memory in a way that (1) preserves base page contiguity and (2) ensures that a large page frame contains pages from only a single memory protection domain. We take advantage of this allocation strategy to design a novel in-place page size selection mechanism that avoids data migration. This mechanism allows the TLB to use large pages, reducing address translation overhead. During data transfer, this mechanism enables the GPU to transfer only the base pages that are needed by the application over the system I/O bus, keeping demand paging overhead low. Our evaluations show that Mosaic reduces address translation overheads while efficiently achieving the benefits of demand paging, compared to a contemporary GPU that uses only a 4KB page size. Relative to a state-of-the-art GPU memory manager, Mosaic improves the performance of homogeneous and heterogeneous multi-application workloads by 55.5% and 29.7% on average, respectively, coming within 6.8% and 15.4% of the performance of an ideal TLB where all TLB requests are hits.

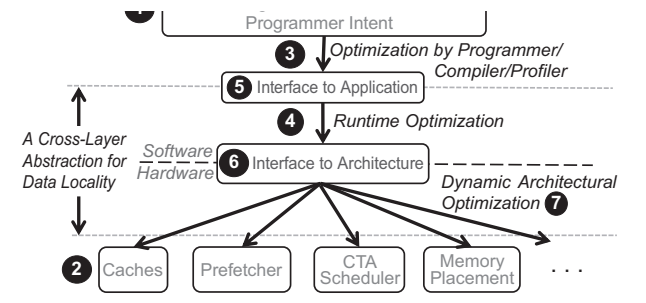
The Locality Descriptor: A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs

Nandita Vijaykumar, Eiman Ebrahimi, Kevin Hsieh, Phillip B. Gibbons & Onur Mutlu

The 45th International Symposium on Computer Architecture - June 2-6, ISCA 2018. Los Angeles, California, USA.

Exploiting data locality in GPUs is critical to making more efficient use of the existing caches and the NUMA-

continued on page 11



Overview of the proposed holistic cross-layer abstraction. The goal is to connect program semantics and programmer intent (1) with the underlying architectural mechanisms (2). By doing so, we enable optimization at different levels of the stack: (i) as an additional knob for static code tuning by the programmer, compiler, or autotuner (3), (ii) runtime software optimization (4), and (iii) dynamic architectural optimization (7) using a combination of architectural techniques. This abstraction interfaces with a parallel GPU programming model like CUDA (5) and conveys key program semantics to the architecture through low overhead interfaces (6).

continued from page 10

based memory hierarchy expected in future GPUs. While modern GPU programming models are designed to explicitly express parallelism, there is no clear explicit way to express data locality—i.e., reuse-based locality to make efficient use of the caches, or NUMA locality to efficiently utilize a NUMA system. On the one hand, this lack of expressiveness makes it a very challenging task for the programmer to write code to get the best performance out of the memory hierarchy. On the other hand, hardware-only architectural techniques are often suboptimal as they miss key higher-level program semantics that are essential to effectively exploit data locality.

In this work, we propose the Locality Descriptor, a crosslayer abstraction to explicitly express and exploit data locality in GPUs. The Locality Descriptor (i) provides the software a flexible and portable interface to optimize for data locality, requiring no knowledge of the underlying memory techniques and resources, and (ii) enables the architecture to leverage key program semantics and effectively coordinate a range of techniques (e.g., CTA scheduling, cache management, memory placement) to exploit locality in a programmer-transparent manner. We demonstrate that the Locality Descriptor improves performance by 26.6% on average (up to 46.6%) when exploiting reuse-based locality in the cache hierarchy, and by 53.7% (up to 2.8X) when exploiting NUMA locality in a NUMA memory system.

Picking Interesting Frames in Streaming Video

Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G. Andersen, Michael Kaminsky & Subramanya R. Dulloor

SysML'18, February 15–16, 2018, Stanford, CA.

As video camera deployments proliferate in the smart cities of the future [2], software systems are faced with the increasing challenge of determining which segments of data are relevant. For resource-constrained edge nodes, limited network bandwidth back to the datacenter prevents sending entire video streams.

This paper presents a new application-independent interesting frame (IF) detection algorithm for identifying relevant frames in streaming video. We envision this IF detector as a preprocessing step in a larger video analytics pipeline where the expensive computation occurs later (similar to the way Bloom filters can guard expensive data structures). Given a target frame rate (or, equivalently, a target bandwidth), the algorithm decides which frames are the most generally interesting and therefore should be processed by downstream applications or forwarded to the datacenter. We decide how "interesting" a frame is based on its semantic difference from other frames. The IF detection algorithm uses a hierarchy of filters to trade off between end-to-end latency and aggressive decimation. The algorithm strives to maximize the semantic diversity of the selected frames. Compared to simply choosing frames at a fixed interval, the IF detector better handles bursty events in the stream.

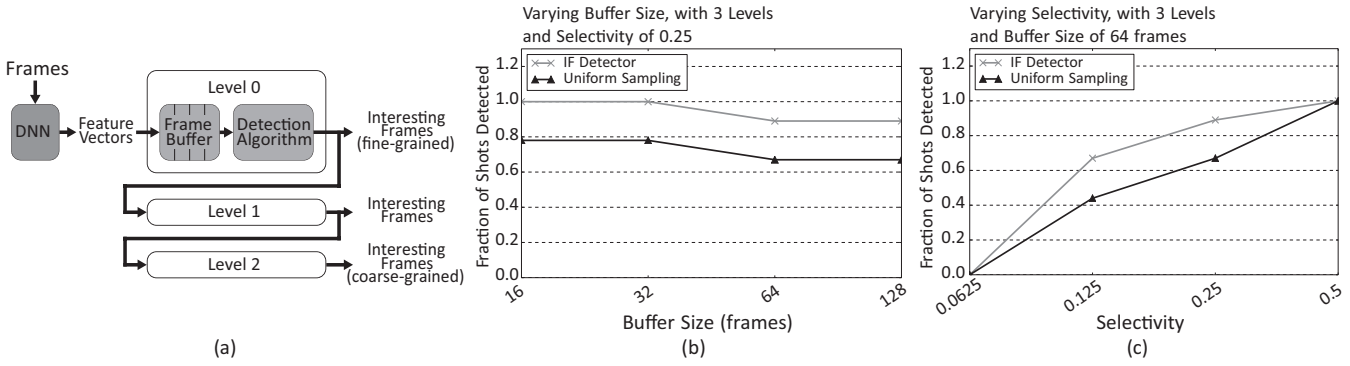
Query-based Workload Forecasting for Self-Driving Database Management Systems

Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo & Geoffrey J. Gordon

SIGMOD/PODS '18 International Conference on Management of Data Houston, TX, USA, June 10-15, 2018.

The first step towards an autonomous database management system (DBMS) is the ability to model the target application's workload. This is necessary to allow the system to anticipate future workload needs and select the proper optimizations in a timely manner. Previous forecasting techniques model the resource utilization of the queries. Such metrics, however, change whenever the physical design of the database

continued on page 12



System architecture and detection accuracy when varying the buffer size and selectivity. Figures b and c demonstrate that for various buffer sizes andselectivities, the IF detector consistently achieves superior shotcoverage compared to uniform sampling.

continued from page 11

and the hardware resources change, thereby rendering previous forecasting models useless.

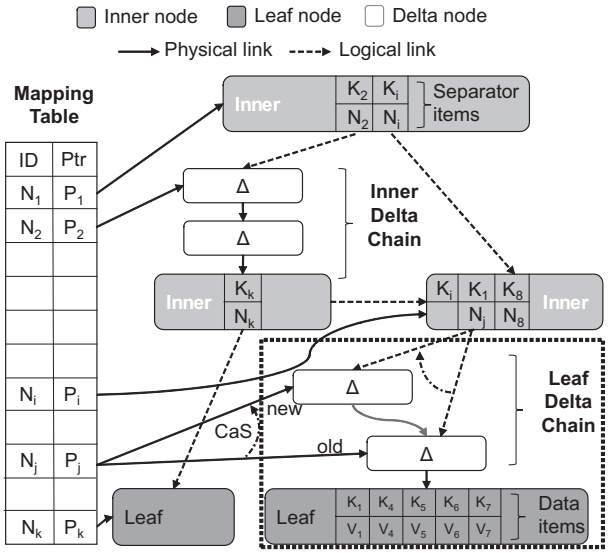
We present a robust forecasting framework called QueryBot 5000 that allows a DBMS to predict the expected arrival rate of queries in the future based on historical data. To better support highly dynamic environments, our approach uses the logical composition of queries in the workload rather than the amount of physical resources used for query execution. It provides multiple horizons (short- vs. long-term) with different aggregation intervals. We also present a clustering-based technique for reducing the total number of forecasting models to maintain. To evaluate our approach, we compare our forecasting models against other state-of-the-art models on three real-world database traces. We implemented our models in an external controller for PostgreSQL and MySQL and demonstrate their effectiveness in selecting indexes.

A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory

Nandita Vijaykumar, Abhilasha Jain, Diptesh Majumdar, Kevin Hsieh, Gennady Pekhimenko, Eiman Ebrahimi, Nastaran Hajinazar, Phillip B. Gibbons & Onur Mutlu

45th International Symposium on Computer Architecture (ISCA), Los Angeles, CA, USA, June 2018.

This paper makes a case for a new cross-layer interface, Expressive Memory (XMem), to communicate higher-level program semantics from the application to the system software and hardware architecture. XMem provides (i) a flexible and extensible abstraction, called an Atom, enabling the application to express key program semantics in terms of how the program accesses data and the attributes of the data itself, and (ii) new cross-layer interfaces to make the expressed higher-level information available to the underlying OS and architecture. By providing key information that is otherwise unavailable, XMem exposes a new, rich view of the program data to the OS and the different architectural components that optimize memory system performance (e.g., caches, memory controllers).



Architecture Overview – An instance of a Bw-Tree with its internal logical links, Mapping Table links, and an ongoing CaS operation on the leaf Delta Chain.

By bridging the semantic gap between the application and the underlying memory resources, XMem provides two key benefits. First, it enables architectural/system-level techniques to leverage key program semantics that are challenging to predict or infer. Second, it improves the efficacy and portability of software optimizations by alleviating the need to tune code for specific hardware resources (e.g., cache space). While XMem is designed to enhance and enable a wide range of memory optimizations, we demonstrate the benefits of XMem using two use cases: (i) improving the performance portability of software-based cache optimization by expressing the semantics of data locality in the optimization and (ii) improving the performance of OS-based page placement in DRAM by leveraging the semantics of data structures and their access properties.

Building a Bw-Tree Takes More Than Just Buzz Words

Ziqi Wang, Andrew Pavlo, Hyeontaek Lim, Viktor Leis, Huanchen Zhang, Michael Kaminsky & David G. Andersen

SIGMOD/PODS '18 International Conference on Management of Data Houston, TX, USA—June 10-15, 2018.

In 2013, Microsoft Research proposed the Bw-Tree (humorously termed the "Buzz Word Tree"), a lock-free index that provides high throughput for transactional database workloads in SQL Server's Hekaton engine. The Bw-Tree avoids locks by appending delta record to tree nodes and using an indirection layer that allows it to atomically update physical pointers using compare-and-swap (CaS). Correctly implementing this techniques requires careful attention to detail. Unfortunately, the Bw-Tree papers from Microsoft are missing important details and the source code has not been released.

This paper has two contributions: First, it is the missing guide for how to build a lock-free Bw-Tree. We clarify missing points in Microsoft's original design documents and then present techniques to improve the index's performance. Although our focus here is on the Bw-Tree, many of our methods apply more broadly to designing and implementing future lock-free in-memory data structures. Our experimental evaluation shows that our optimized variant achieves

continued on page 13

continued from page 12

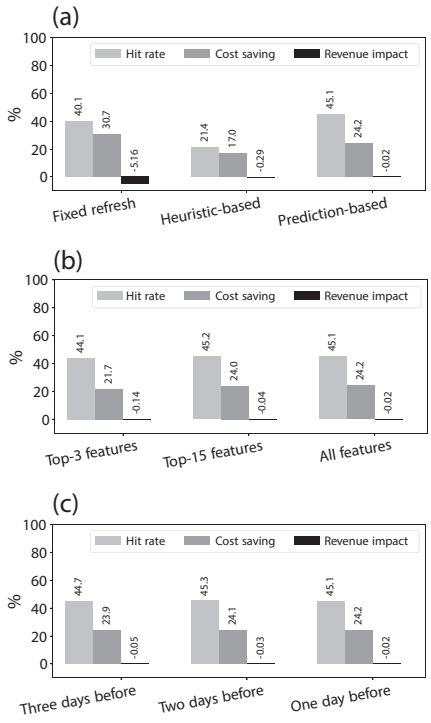
1.1–2.5× better performance than the original Microsoft proposal for highly concurrent workloads. Second, our evaluation shows that despite our improvements, the Bw-Tree still does not perform as well as other concurrent data structures that use locks.

Better Caching in Search Advertising Systems with Rapid Refresh Predictions

Conglong Li, David G Andersen, Qiang Fu, Sameh Elnikety & Yuxiong He

Proceedings of the 2018 World Wide Web Conference on World Wide Web.

To maximize profit and connect users to relevant products and services, search advertising systems use sophisticated machine learning algorithms to estimate the revenue expectations of thousands of matching ad listings per query. These machine learning computations constitute a substantial part of the operating cost, e.g., 10% to 30% of the total gross revenues. It is desirable to cache and reuse previous computation results to reduce this cost, but caching introduces approximation which comes with potential revenue loss. To maximize cost savings while minimizing the overall revenue impact, an intelligent refresh policy is required to decide when to refresh the cached computation results. The state-of-the-art manually-tuned refresh heuristic uses revenue history to assign different refresh frequencies. Using the gradient boosting regression tree algorithm with well selected features, we introduce a rapid prediction framework that provides refresh decisions at higher accuracy compared to the heuristic. This enables us to build a prediction-based refresh policy and a cache achieving higher profit without manual parameter tuning. Simulations conducted on the logs from a major commercial search advertising system show that our proposed cache design reduces the negative revenue impact (0.07x), and improves the cost savings (1.41x) and the net profit (1.50~1.70x) compared to the state-of-the-art manually-tuned heuristic-based cache design.



(a) Hit rate, cost saving, and revenue impact for naive fixed refresh rate, heuristic-based cache, and proposed prediction-based cache. For all the numbers the higher the better. (b) Hit rate, cost saving, and revenue impact for prediction-based cache with different feature sets. The corresponding net profit impacts are: $29.8 to $93.4 million, $34.6 to $105.0 million, and $35.2 to $106.1 million. (c) Hit rate, cost saving, and revenue impact for prediction-based cache with different training data. The corresponding net profit impacts are: $34.3 to $104.2 million, $34.9 to $105.5 million, and $35.2 to $106.1 million.

Practical Bounds on Optimal Caching with Variable Object Sizes

Daniel S. Berger, Nathan Beckmann & Mor Harchol-Balter

ACM Proceedings on Measurement and Analysis of Computing Systems. Vol. 2, No. 2, Article 32. June 2018.

Many recent caching systems aim to improve miss ratios, but there is no good sense among practitioners of how much further miss ratios can be improved. In other words, should the systems community continue working on this problem?

Currently, there is no principled answer to this question. In practice, object sizes often vary by several orders of magnitude, where computing the optimal miss ratio (OPT) is known to be NP-hard. The few known results on caching with variable object sizes provide very weak bounds and are impractical to compute on traces of realistic length.

We propose a new method to compute upper and lower bounds on OPT. Our key insight is to represent caching as a min-cost flow problem, hence we call our method the flow-based offline optimal (FOO). We prove that, under simple independence assumptions, FOO's bounds become tight as the number of objects goes to infinity. Indeed, FOO's error over 10M requests of production CDN and storage traces is negligible: at most 0.3%. FOO thus reveals, for the first time, the limits of caching with variable object sizes. While FOO is very accurate, it is computationally impractical on traces with hundreds of millions of requests. We therefore extend FOO to obtain more efficient bounds on OPT, which we call practical flow-based offline optimal (PFOO).We evaluate PFOO on several full production traces and use it to compare OPT to prior online policies. This analysis shows that current caching systems are in fact still far from optimal, suffering 11–43% more cache misses than OPT, whereas the best prior offline bounds suggest that there is essentially no room for improvement.

continued on page 14

continued from page 13

Implicit Decomposition for Write-Efficient Connectivity Algorithms

Naama Ben-David, Guy E. Blelloch, Jeremy T. Fineman, Phillip B. Gibbons, Yan Gu, Charles McGuffey & Julian Shun

2018 International Parallel and Distributed Processing Symposium (IPDPS '18). May 21-25, 2018, Vancouver, BC, Canada.

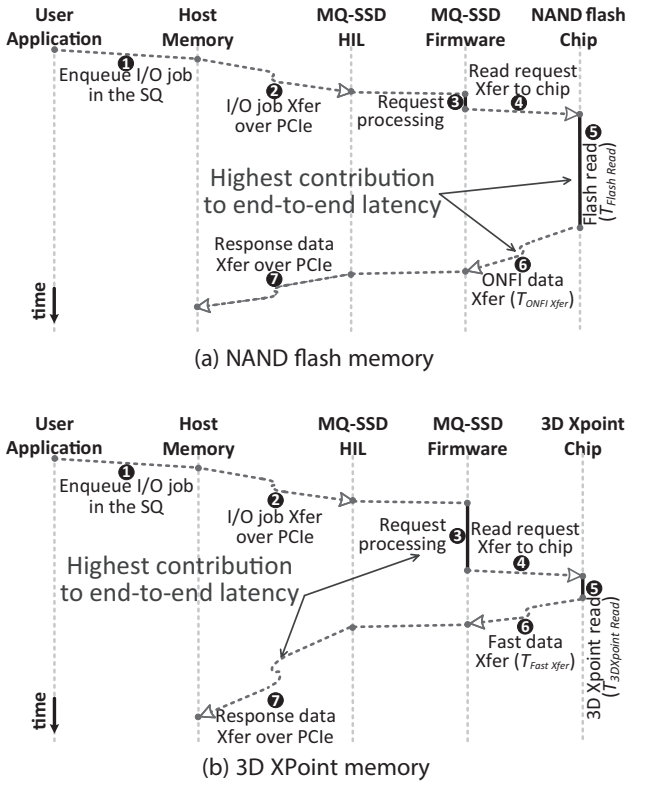
The future of main memory appears to lie in the direction of new technologies that provide strong capacity-to-performance ratios, but have write operations that are much more expensive than reads in terms of latency, bandwidth, and energy. Motivated by this trend, we propose sequential and parallel algorithms to solve graph connectivity problems using significantly fewer writes than conventional algorithms. Our primary algorithmic tool is the construction of an o(n)-sized implicit decomposition of a bounded-degree graph G, which combined with read-only access to G enables fast answers to connectivity and biconnectivity queries on G. The construction breaks the linear-write "barrier", resulting in costs that are asymptotically lower than conventional algorithms while adding only a modest cost to querying time. For general non-sparse graphs, we also provide the first o(m) writes and O(m) operations parallel algorithms for connectivity and biconnectivity. These algorithms provide insight into how applications can efficiently process computations on large graphs in systems with read-write asymmetry.

MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices

Arash Tavakkol, Juan Gómez-Luna, Mohammad Sadrosadati, Saugata Ghose & Onur Mutlu

USENIX Conference on File and Storage Technologies (FAST), Oakland, CA, February 2018.

Solid-state drives (SSDs) are used in a wide array of computer systems today, including in datacenters and enterprise servers. As the I/O demands of these systems continue to increase, manufacturers are evolving SSD architectures to keep up with this demand. For example, manufacturers have introduced new high-bandwidth interfaces to replace the conventional SATA host–interface protocol. These new interfaces, such as the NVMe protocol, are designed specifically to enable the high amounts of concurrent I/O bandwidth that SSDs are capable of delivering.



Timing diagram for a 4 kB read request in (a) NAND-flash and (b) 3D XPoint MQ-SSDs.

While modern SSDs with sophisticated features such as the NVMe protocol are already on the market, existing SSD simulation tools have fallen behind, as they do not capture these new features. We find that state-of-theart SSD simulators have three shortcomings that prevent them from accurately modeling the performance of real off-the-shelf SSDs. First, these simulators do not model critical features of new protocols (e.g., NVMe), such as their use of multiple application-level queues for requests and the elimination of OS intervention for I/O request processing. Second, these simulators often do not accurately capture the impact of advanced SSD maintenance algorithms (e.g., garbage collection), as they do not properly or quickly emulate steady-state conditions that can significantly change the behavior of these algorithms in real SSDs. Third, these simulators do not capture the full end-to-end latency of I/O requests, which can incorrectly skew the results reported for SSDs that make use of emerging non-volatile memory technologies. By not accurately modeling these three features, existing simulators report results that deviate significantly from real SSD performance.

In this work, we introduce a new simulator, called MQSim, that accurately models the performance of both modern SSDs and conventional SATA-based SSDs. MQSim faithfully models new high-bandwidth protocol implementations, steady-state SSD conditions, and the full end-to-end

continued on page 15

continued from page 14

latency of requests in modern SSDs. We validate MQSim, showing that it reports performance results that are only 6%-18% apart from the measured actual performance of four real state-of-the-art SSDs. We show that by modeling critical features of modern SSDs, MQSim uncovers several real and important issues that were not captured by existing simulators, such as the performance impact of inter-flow interference. We have released MQSim as an open-source tool, and we hope that it can enable researchers to explore directions in new and different areas.

Rateless Codes for Near-Perfect Load Balancing in Distributed Matrix-Vector Multiplication

Ankur Mallick, Malhar Chaudhari & Gauri Joshi

arXiv:1804.10331v2 [cs.DC] 30 Apr 2018

Large-scale machine learning and data mining applications require computer systems to perform massive computations that need to be parallelized across multiple nodes, for example, massive matrix-vector and matrix-matrix multiplication. The presence of straggling nodes – computing nodes that unpredictably slowdown or fail – is a major bottleneck in such distributed computations. We propose a rateless fountain coding strategy to alleviate the problem of stragglers in distributed matrix-vector multiplication. Our algorithm creates a stream of linear combinations of the m rows of the matrix, and assigns them to different worker nodes, which then perform row-vector products with the encoded rows. The original matrix-vector product can be decoded as soon as slightly more than m row-vector products are collectively finished by the nodes. This strategy enables fast nodes to steal work from slow nodes, without requiring the master to perform any dynamic load-balancing. Compared to recently proposed fixed-rate erasure coding strategies which ignore partial work done by straggling nodes, rateless coding achieves significantly lower overall delay, as well as small computational and decoding overhead.

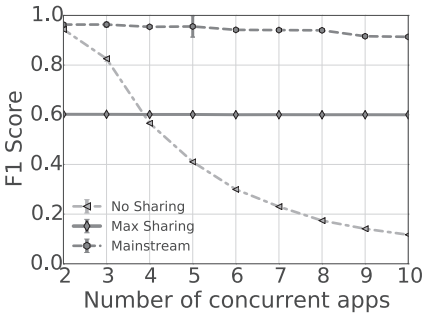
Dynamic Stem-Sharing for Multi-Tenant Video Processing

Angela Jiang, Christopher Canel, Daniel Wong, Michael Kaminsky, Michael A. Kozuch, Padmanabhan Pillai, David G. Andersen & Gregory R. Ganger

SysML 18, February 15–16, 2018. Stanford, CA.

Video cameras are ubiquitous, and their outputs are increasingly analyzed by sophisticated, online DNN inference-based applications. The ever-growing capabilities of video and image analysis techniques create new possibilities for what may be gleaned from any given video stream. Consequently, most raw video streams will be processed by multiple analysis pipelines. For example, a parking lot camera might be used by three different applications: reporting open parking spots, tracking each car's parking duration for billing, and recording any fender benders.

In this paper, we focus on shared processing on edge devices; processing video near the camera addresses issues such as bandwidth, intermittent connectivity (e.g., in drones), and real-time requirements, but leads to resource limitations. Thus, optimal video application performance requires tuning to the resources available [13, 2, 14, 4, 7]. However, application developers may be unable to predict easily what resources will be available when the application is deployed, particularly in "multi-tenant" environments where the set of concurrently deployed applications may vary. Instead, individual application developers typically develop their models in isolation, assuming either infinite resources or a predetermined set of static resources. When a number of such individually-tailored models are run concurrently, resource competition forces the video stream to be analyzed at a lower frame rate— leading to unsatisfactory results for the running applications, as frames are dropped and events in those frames are missed.



Mainstream improves application quality (average F1- score) relative to both (a) no sharing between applications (Nosharing) and (b) sharing all layers but the last one (Max-sharing).

Garth Gibson talks about his 25 years with the PDL at the 2018 PDL Retreat. Photo credit, Allyson Lowe.

The Mainstream video processing system enables efficient execution of multiple independently-developed and incrementally-deployed video analysis applications on a given video stream. Mainstream shares execution of concurrent DNNs, yet does not rely on applications' DNNs to be trained collectively. Therefore, Mainstream provides collaborative execution, even when development and training data are not centralized in one organization.

continued on page 16

RECENT PUBLICATIONS

continued from page 15

Efficient Multi-Tenant Inference on Video using Microclassifiers

Giulio Zhou, Thomas Kim, Christopher Canel, Conglong Li, Hyeontaek Lim, David G. Andersen, Michael Kaminsky & Subramanya R. Dulloor

SysML'18, February 15–16, 2018, Stanford, CA.

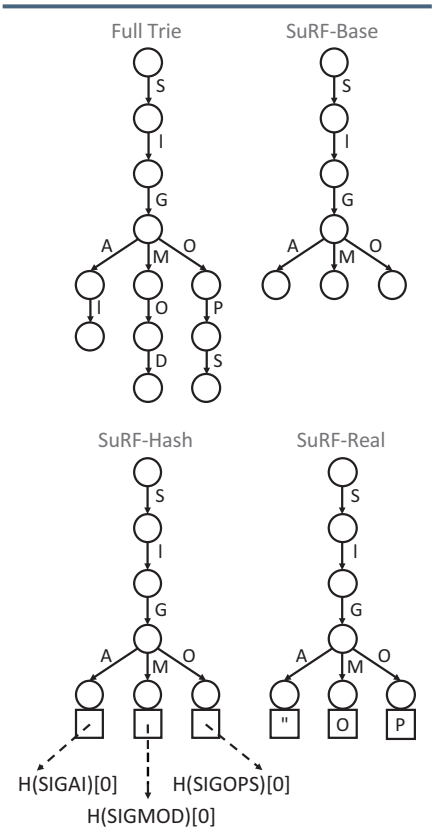
This paper addresses a growing challenge in processing video: The scaling challenge presented by the combination of an increasing number of video sources (cameras) and an increasing number of heavy-weight DNN-based applications (which we term "queries") to be run on each source. As a running example, we draw from an environmental and traffic monitoring deployment at CMU, one feed from which is depicted at right. This feed supports applications such as car and pedestrian counting, open parking spot detection, train detection (in support of an environmental monitoring research project attempting to determine locomotive emissions), and observing if building lights are left on. These cameras are deployed using a mix of the high-speed campus network, and a lower-speed/higher-cost cable modem deployment on power poles in the area.

SuRF: Practical Range Query Filtering with Fast Succinct Tries

Huanchen Zhang, Hyeontaek Lim, Viktor Leis, David G. Andersen, Michael Kaminsky, Kimberly Keeton & Andrew Pavlo

SIGMOD'18, June 10–15, 2018, Houston, TX, USA.

We present the Succinct Range Filter (SuRF), a fast and compact data structure for approximate membership tests. Unlike traditional Bloom filters, SuRF supports both single-key lookups and common range queries: open-range queries, closed-range queries, and range counts. SuRF is based on a new data structure called the Fast Succinct Trie (FST) that matches the point and range query performance of state-of-the-art order-preserving indexes, while consuming only 10 bits per trie node. The false positive rates in SuRF for both point and range queries are tunable to satisfy different application needs. We evaluate SuRF in RocksDB as a replacement for its Bloom filters to reduce I/O by filtering requests before they access on-disk data structures. Our experiments on a 100 GB dataset show that replacing RocksDB's Bloom filters with SuRFs speeds up open-seek (without upper-bound) and closed-seek (with upper-bound) queries by up to 1.5× and 5× with a modest cost on the worst-case (all-missing) point query throughput due to slightly higher false positive rate.



An example of deriving SuRF variations from a full trie.

2017 PDL Workshop and Retreat.