

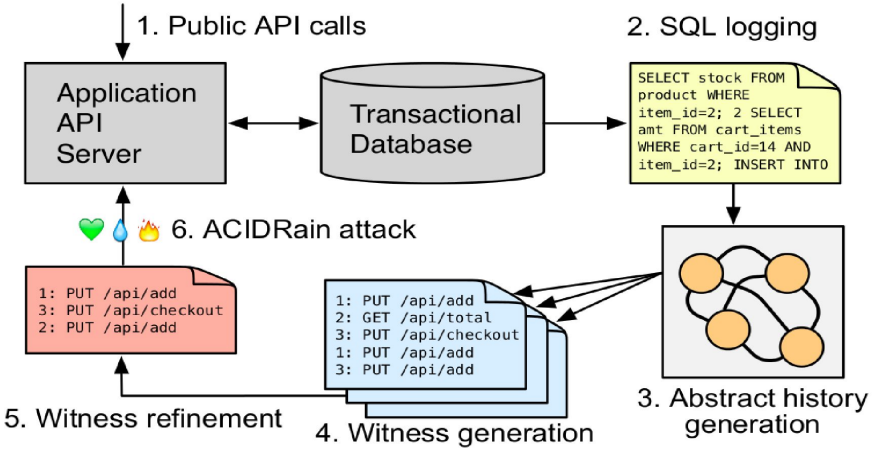
SIGMOD 2017 Teaser Talks

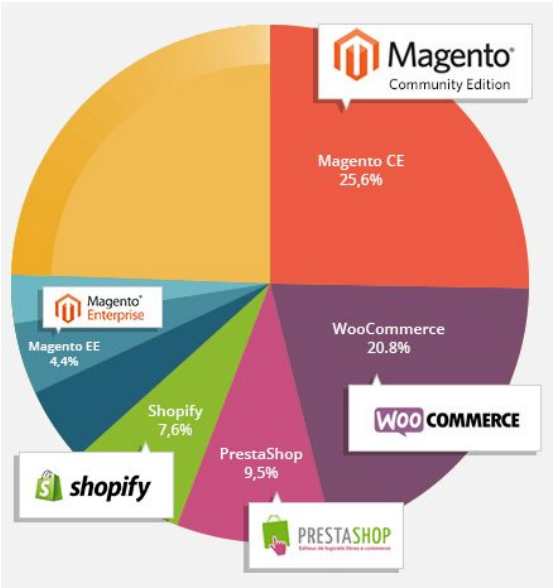
Tuesday - May 17th 2017

ACIDRain: Concurrency-Related Attacks on Database-Backed Web Applications

Todd Warszawski and Peter Bailis (Stanford University)

- Do programmers use transactions correctly?
- Analyzed 12 open source eCommerce applications
- 22 new vulnerabilities
- 2M+ websites (50% of eCommerce sites) at risk





Cicada: Dependably Fast Multi-Core In-Memory Transactions

Concurrency

Hyeontaek Lim, Michael Kaminsky, David G. Andersen (CMU & Intel Labs)





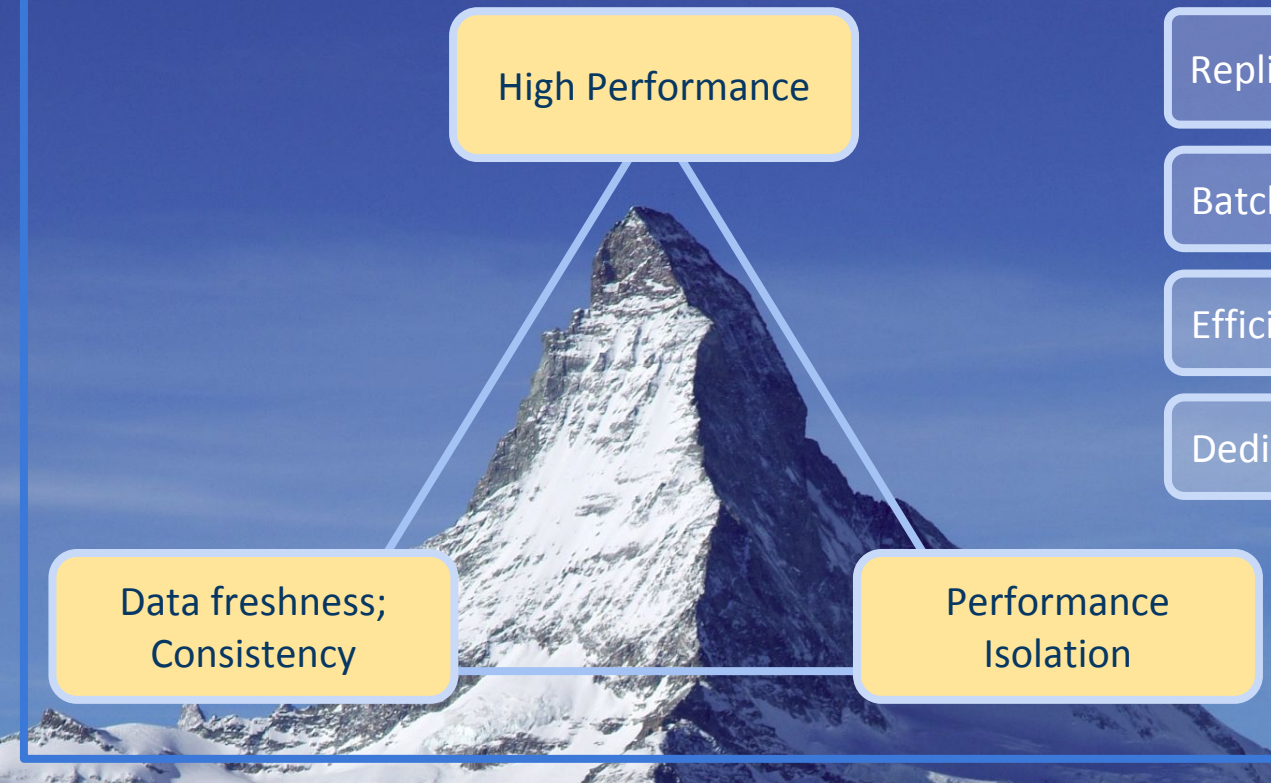
You need Cicada. Dependably Fast Transactions.

See us at 11:25 in Concurrency (1), Continental B.

Concurrency

BatchDB: Efficient Isolated Execution of Hybrid OLTP+OLAP Workloads

D. Makreshanski, J. Giceva, C. Barthels, & G. Alonso (ETH Zurich)

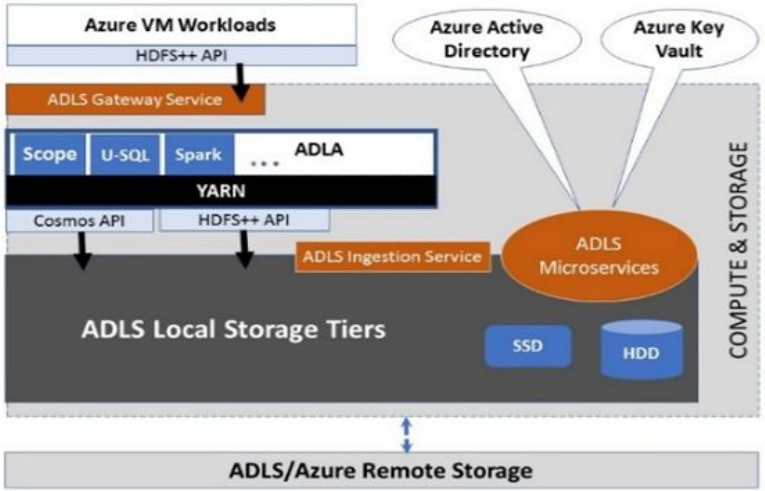


- Replication
- Batch Scheduling
- Efficient Update Propagation
- Dedicated resources

Azure Data Lake Store

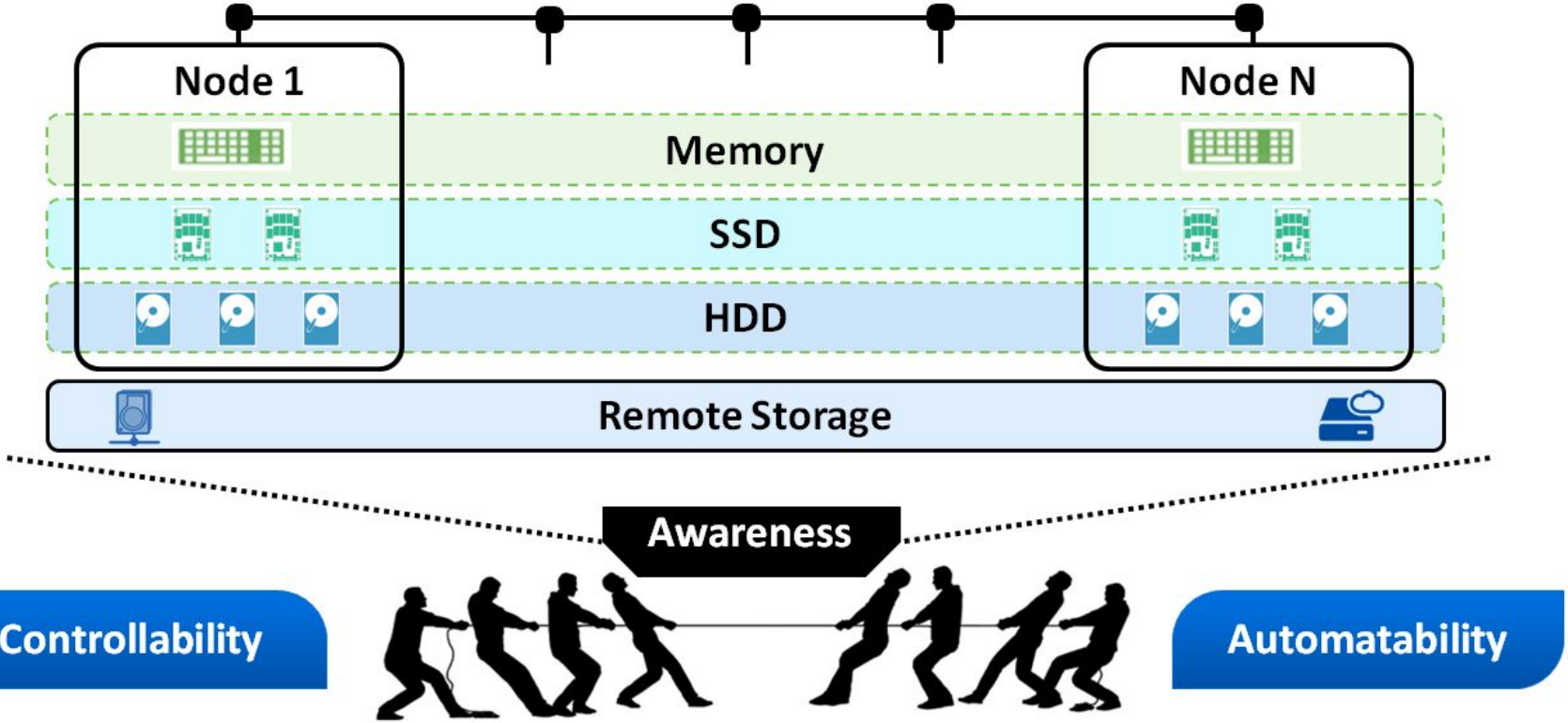
Raghu Ramakrishnan and others (Microsoft)

- A hyper-scale, reliable cloud file system for analytics based on a decade of experience with Cosmos (Microsoft's internal warehouse for data and analytics)
  - Individual files of 1PB or more easily handled
  - Currently storing billions of files
- Unified storage solution for workloads needing a very high degree of parallel reads and writes
- Single system supports batch, streaming and interactive workloads
- Support for open source analytic engines and rest of big data stack (including Hadoop and Spark) via open APIs
- Fast, scalable, resilient filesystem metadata services built on top of a replicated, in-memory NoSQL database
- Naming service provides true file system semantics with real file and folder hierarchy
- Security via POSIX ACLs and encryption at rest



OctopusFS: A Distributed File System with Tiered Storage Management

Elena Kakoulli & **Herodotos Herodotou** (Cyprus University of Technology)



Monkey: Optimal Navigable Key-Value Store

Niv Dayan, Manos Athanassoulis, Stratos Idreos

Storage & Dist

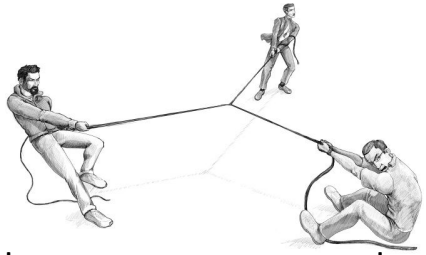
DASlab @ Harvard SEAS



Bloom filters



LSM-tree merging

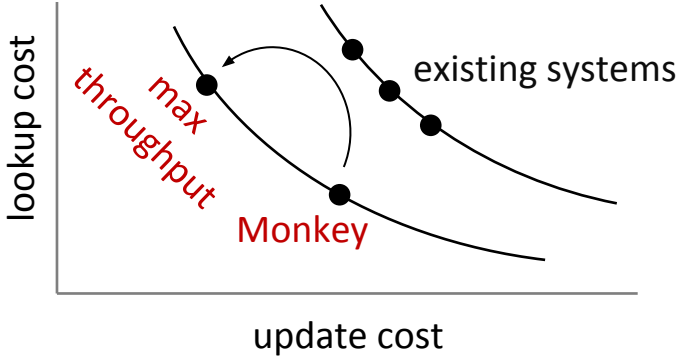


lookup cost update cost

allocation optimization

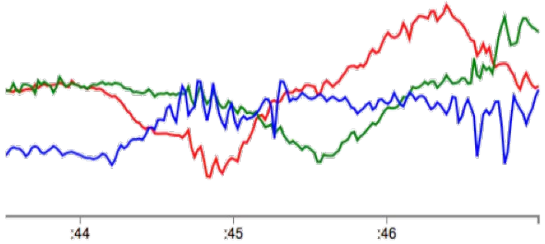


navigation



Enabling Signal Processing over Data Streams

M. Nikolic, B. Chandramouli, & J. Goldstein (U. Oxford & MSR)



- Lots of “signals” in stream data
  - IoT devices, application telemetry
- Apps mix relational & signal logic
  - Per device: find periodicity in signals, interpolate missing data, recover noisy data
  - Different data models: relational vs. array
- Existing query processors integrated with R
  - Impedance mismatch → high performance overhead → not suitable for real-time
- TrillDSP = Relational processing + Signal processing
  - Unified query model for relational and signal data, for both real-time and offline
  - High performance: 2 OOM faster than engines integrated with R

Complete Event Trend Detection in High-Rate Event Streams

O.Poppe*, C.Lei**, S.Ahmed*, E.Rundensteiner* (WPI*, NEC Labs America**)

Application: **Check Kiting Fraud**

Event trend = sequence of any length of bad checks (check kite)



Goal: Real-time event trend detection

Challenge: Exponential number of trends



Streams

LittleTable: A Time-Series Database and Its Uses

Sean Rhea, Eric Wang, Edmund Wong, Ethan Atkins, and Nat Storer

The relational database for all of Cisco Meraki's *time-series data*:

- Usage counters, wireless (dis)association logs, video motion vectors, etc.

Tuned to applications of time-series data (e.g., Meraki Dashboard, IoT apps):

- Insatiable storage needs: always want more resolution, longer retention
- Highly skewed read patterns: most reads to newest data, yet minority read old
- Rows represent facts measured at particular times: no need for updates
- Single writer per source: no read-write consistency concerns
- Recently written data is recoverable: on crash, re-read from original source

Implements log-structured, two-dimensional clustering to get high performance:

- 400k+ rows/sec read/write on a single spinning disk and CPU

Incremental View Maintenance over Array Data

Weijie Zhao, F. Rusu, B. Dong, K. Wu, P. Nugent (UC Merced & LBNL)

Rare Supernova Discovery Ushers in New Era for Cosmology

Berkeley Lab astrophysicists develop nov
supernovae

News Release Linda Vu (510) 495-2402 • Al

'Major breakthrough' as supernova four billion light years from Earth is captured from four different angles in a stunning world first

- Strange event occurred because light from the supernova bent through a galaxy
- Galaxies bend light through an effect that is called 'gravitational lensing'
- The galaxy magnified the supernova 50 times to give astronomers a unique view

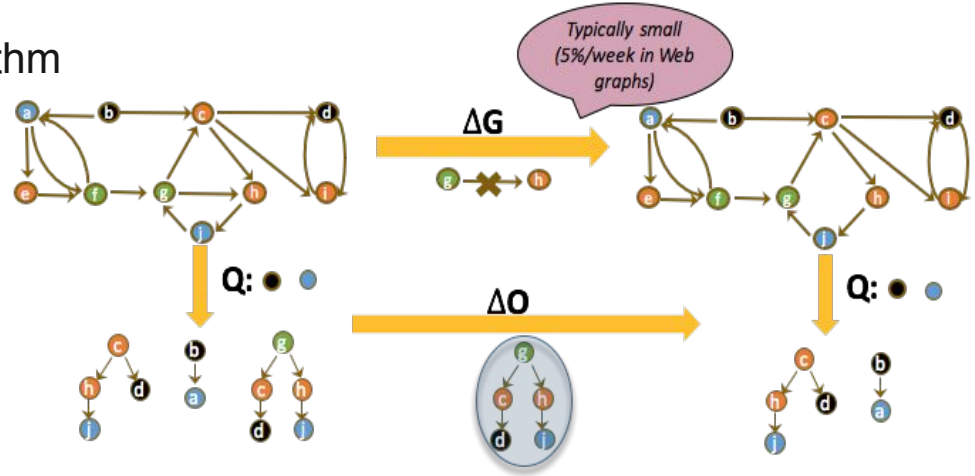
Astronomers Get Rare View Of Type Ia Supernova Magnified 50 Times

21 April 2017, 9:01 am EDT By Luan Chan Tech Times

Incremental Graph Computations: Doable and Undoable

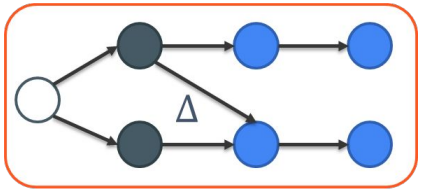
W. Fan, C. Hu, & C. Tian (Univ. of Edinburgh & Beihang Univ.)

- Batch algorithm vs. Incremental Algorithm
- Boundedness - independent of |G|
- Undoable
  - Δ-reductions - a systematic method
  - Unboundedness
    - regular path queries
    - strongly connected components
    - keyword search
- Doable
  - Two new measures: locality and relatively boundedness
  - Localizable incremental computation
    - Keyword search and subgraph isomorphism
  - Relatively bounded incrementalization
    - Regular path queries and keyword search



DEX: Query Execution in a Delta-based Storage System

Amit Chavan | Amol Deshpande
University of Maryland, College Park



Delta Storage

- Common method to store immutable copies of data, e.g., dataset versions
- Existing systems focus on retrieval performance only
- Query:
  Find common records in `file_X` and `file_Y` ?

☹ ∩ `Checkout[file_X]` `Checkout[file_Y]`

Goal:

- Initiate a systematic study of how to execute queries in a delta-based storage system
- Queries: Checkout, Intersection, Union, t-Threshold

Idea:

- Exploit Δ properties & introduce new transformation rules
- Cost-based optimization algorithms based on above

Result:

- Order of magnitude improvements, even for simple checkout queries
- DEX is a wrapper over git and built for dataset versioning

Massively Parallel Processing of Whole Genome Sequence Data: An In-Depth Performance Study

Parallel/Dist QP

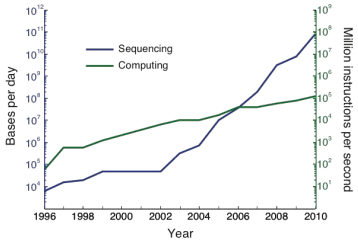
A. Roy, Y. Diao, U. Evani, A. Abhyankar, C. Howarth, R. Le Priol, T. Bloom

(UMass Amherst, New York Genome Center, École Polytechnique)

Sequencing technology has advanced significantly

Amount of sequence data has doubled every seven months, far exceeding Moore's Law

- Sequencing center: ≥10 terabytes a day
- Per sample: 0.5 – 1 terabyte



A new paradigm of data-intensive computing

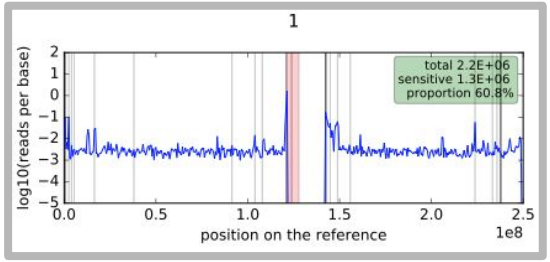
- Answer biomedical questions with high speed and resolution
- Personalized Genomics:

Early Detection | Precise & Specific | No side effects

Objectives

1. A big data platform to reduce processing time from weeks to 1-2 days
2. Strengths and limitations of big data technology and its future research

- *When does big data technology offer superlinear speedup?*
- *When does it provide sublinear speedup?*
- *What is the quality of results (genome mutations)?*

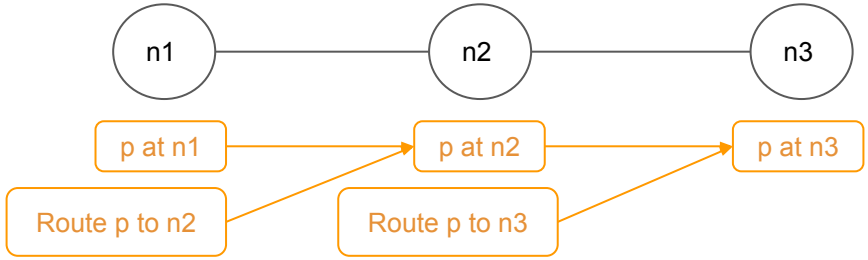


Distributed Provenance Compression

Parallel/Dist QP

Chen Chen*, Harshal Tushar Lehri*, Lay Kuan Loh+, Anupam Alur*, Limin Jia+,
Boon Thau Loo*, Wenchao Zhou# (U. Penn*, CMU+, Georgetown Univ.#)

Network provenance:
Execution history of network events



Problem: high storage overhead

Challenge: reduce storage with low network overhead

Solution: equivalence-based, input-oriented provenance compression

- Group provenance into equivalence classes
- Maintain one copy for each equivalence class

Significant storage reduction

- Identify equivalence based on input attributes
- Avoid generating redundant provenance

Negligible network overhead

Parallel/Dist QP

ROBUS: Fair Cache Allocation for Data-parallel Workloads

Mayuresh Kunjir, Brandon Fain, Kamesh Munagala, Shivnath Babu (Duke University)

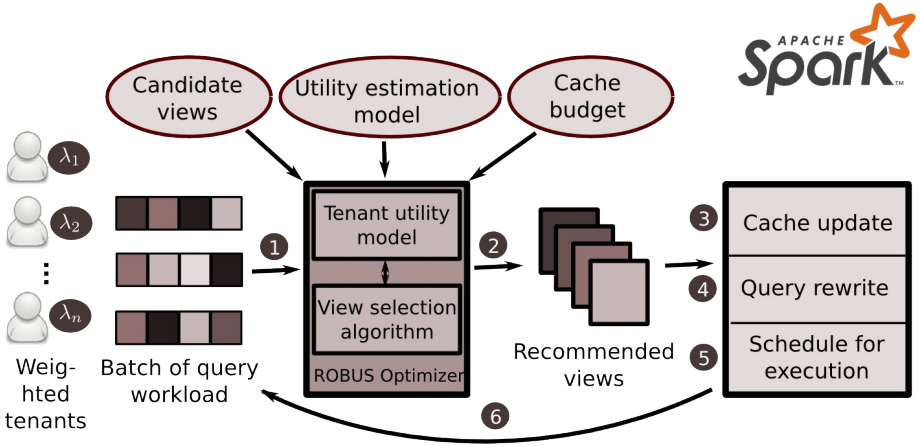
- Cache can lead to performance speedups.
- Cache can be shared by multiple tenants.

Challenge:

How to bring *fairness* to cache allocation?

ROBUS

- Randomized policies
- Optimize for speedup and fairness together
- Batched processing of workload
- Utility modeling
- Sharing aware

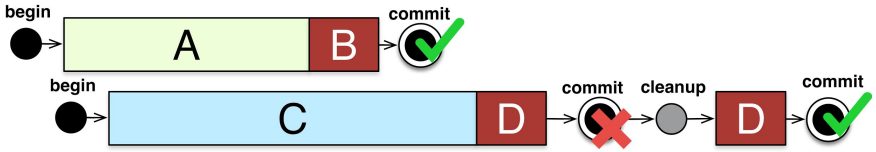


Teaser Talks (Second Part)

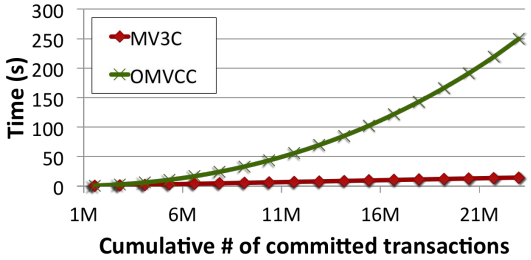
Transaction Repair for Multi-Version Concurrency Control

M. Dashti, S. Basil John, A. Shaikhha, C. Koch (EPFL)

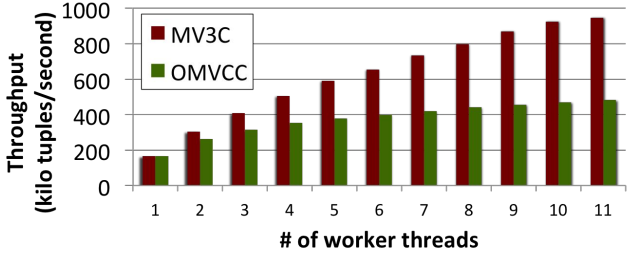
- **Goal:** Reuse computation instead of "abort and restart"
- **Use-Cases:**
  - High-contention objects
  - Long-running transactions



- **Key ideas:**
  - Expose program dependencies to concurrency control algorithm
  - Associate correctness checks used in validation with blocks of code
- **Experimental Results:**



(Banking benchmark)

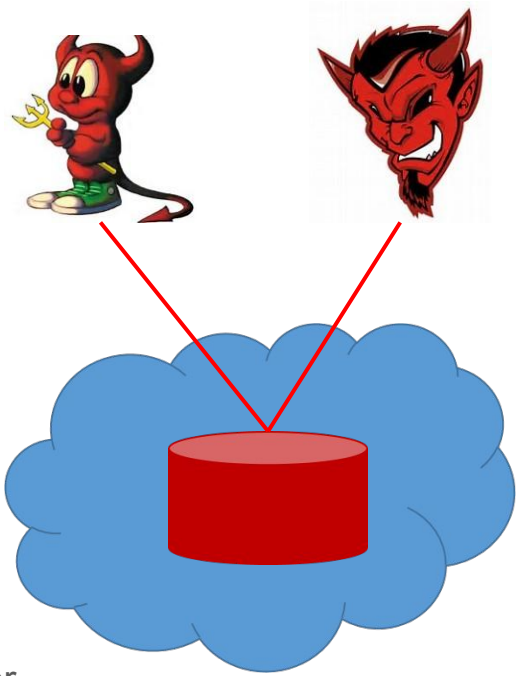


(Trading benchmark)

Concerto in a Nutshell

A. Arasu, K. Eguro, R. Kaushik, D. Kossmann, P. Meng, V. Pandey, & R. Ramamurthy

Microsoft Research, UCSB

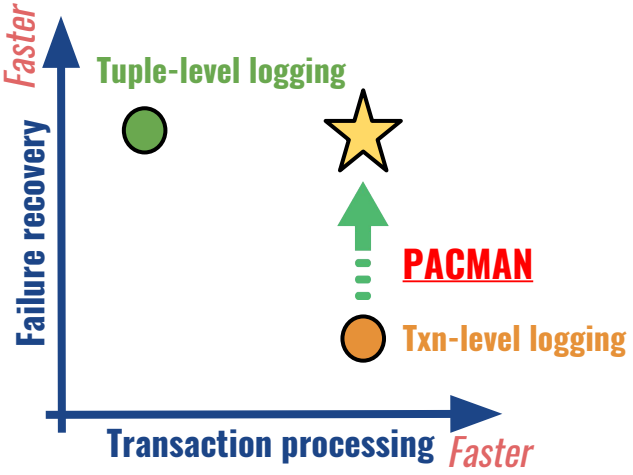


- Goal: People collaborate in the cloud without trust
  - Every update recorded in a tamper-proof way
  - (promise of Blockchain, but in the cloud)
- Traditional approach: Merkle Trees (pessimistic)
  - Provably correct, but slow and poor concurrency
  - 1000s of operations / second
- Our approach: Verified Memory (optimistic)
  - Provably correct, high concurrency, but catches violations later
  - 1,000,000s of operations / second

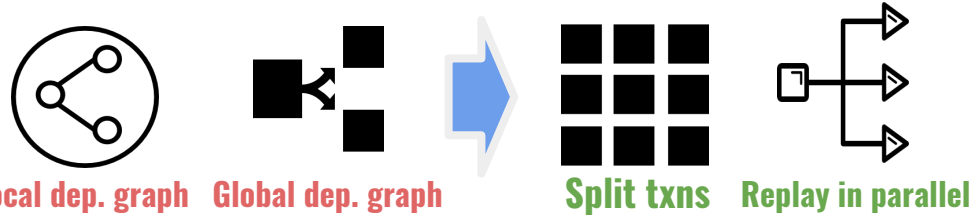
Fast Failure Recovery for Main-Memory DBMSs on Multicores

Concurrency

Y. Wu, W. Guo, C.-Y. Chan, K.-L. Tan (National University of Singapore)



- Static Time
  - Construct local dependency graphs
  - Construct global dependency graph
- Recovery Time
  - Split transactions into pieces
  - Replay transaction pieces



PACMAN is a parallel log recovery approach designed for transaction-level logging.

Key idea: leverage a combination of static and dynamic analysis

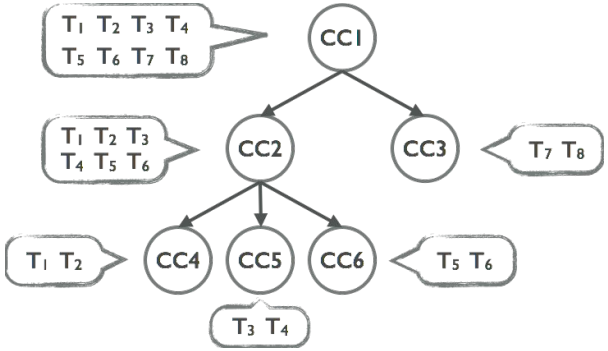
PACMAN achieves efficient failure recovery without compromising transaction processing performance!

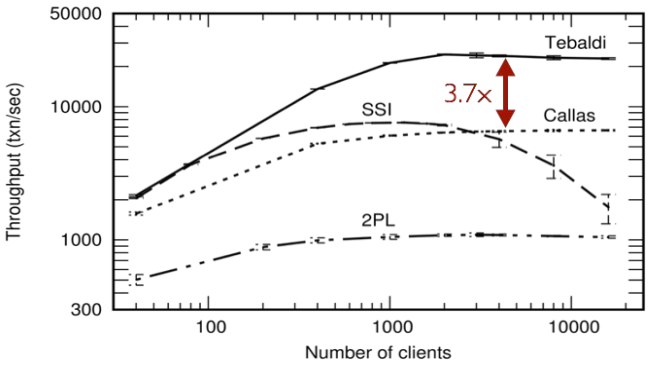
Bringing Modular Concurrency Control to the Next Level

C. Su, N. Crooks, C. Ding, L. Alvisi, C. Xie (UT Austin & Cornell)

Tebaldi: a distributed transactional key-value store that federates multiple CCs

- Motivation & Approach:
  - No single concurrency control fits all workloads.
  - Partition txns, assign each group a specialized CC.
  - **But conflicts across groups matter!**
- Our Secret Sauce:
  - Organize CCs as nodes in a **multi-level tree**.
  - Each node manages conflicts for a set of txns.
  - Nodes delegate responsibility for disjoint subsets of txns to children better-suited to handle them.
- Results:
  - 3.7x higher throughput than Callas [SOSP ‘15].

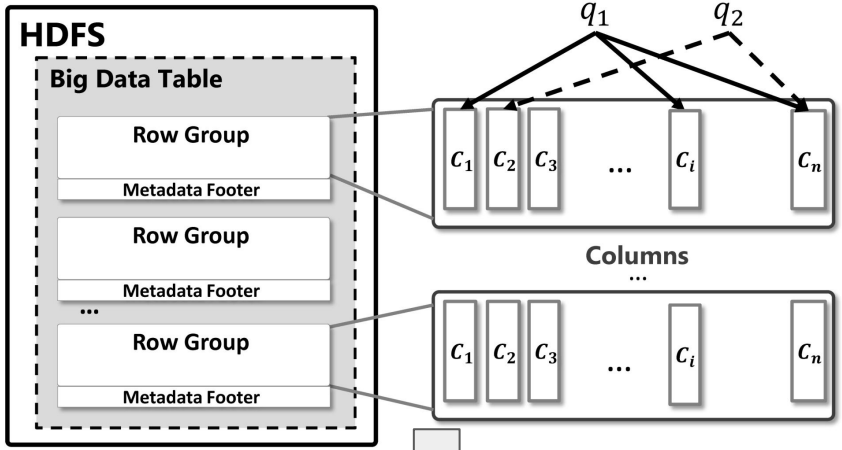


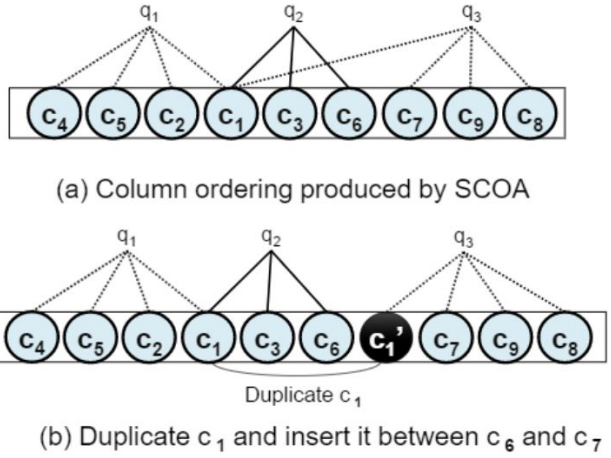


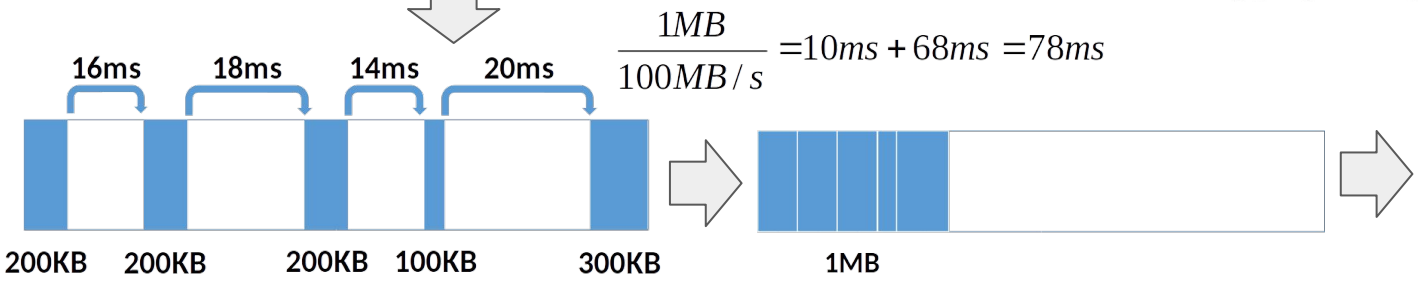
Wide Table Layout Optimization based on Column Ordering and Duplication

H. Bian, Y. Yan, W. Tao, L. Chen, Y. Chen, X. Du, T. Moscibroda

(Renmin U. of China, MSR, & MIT)





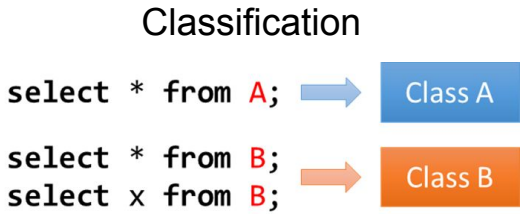


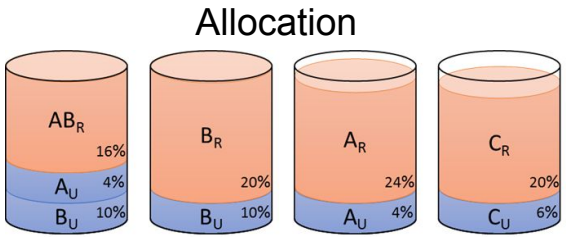
1000+Columns
4000+Queries
How to make a BEST column order?

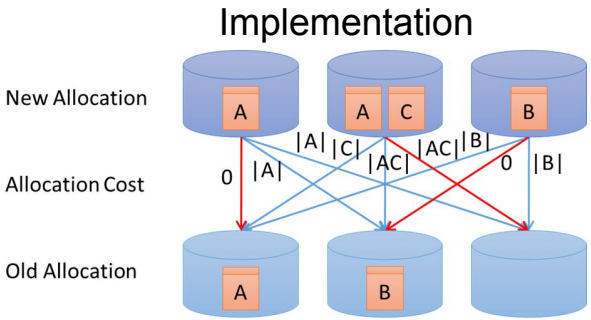
Query Centric Partitioning and Allocation for Partially Replicated Database Systems

Tilmann Rabl & Hans-Arno Jacobsen (TU Berlin)

Analytical model and automatic strategy for query centric partitioning and allocation







Key benefits:

- Predictable performance
- Robustness
- Linear (read), good (write) speedup







Storage & Dist

Spanner: Becoming a SQL System

D. Bacon, N. Bales, N. Bruno, B. Cooper, A. Dickinson, A. Fikes, C. Fraser, A. Gubarev, M. Joshi, E. Kogan, A. Lloyd, Sergey Melnik, C. Taylor, R. Rao, D. Shue, M. van der Holst, D. Woodford (Google)

- Distributed transactional data management system
- Globally replicated, highly-available managed service
- Backs hundreds of mission-critical services at Google
  - AdWords, Google Play, Photos, etc.
  - 10s of millions QPS, 100s of petabytes, 5,000+ databases
- Publicly available on Google Cloud Platform: http://cloud.google.com/spanner
- This talk: making Spanner a SQL DBMS

Session 7: Storage and Distribution (2)
14:00-15:40 @ Continental C

Tree & Graph

Landmark indexing for evaluation of label-constrained reachability queries

L. Valstar, G. Fletcher (TU Eindhoven, Netherlands),
Y. Yoshida (National Institute of Informatics, Japan & Preferred Infrastructure, Inc.)

In a social network, are Jane and John connected by a chain of social relationships?

In a biological network, is there an interaction pathway between two particular proteins?

Unfortunately, current solutions for evaluating such “label-constrained reachability queries” do not scale to massive graphs occurring in practice.

Our contributions: new indexing methods for efficient LCR evaluation,

- scaling to orders of magnitude larger graphs than current LCR indexing strategies,
- with up to orders of magnitude faster query evaluation than state of the art methods.

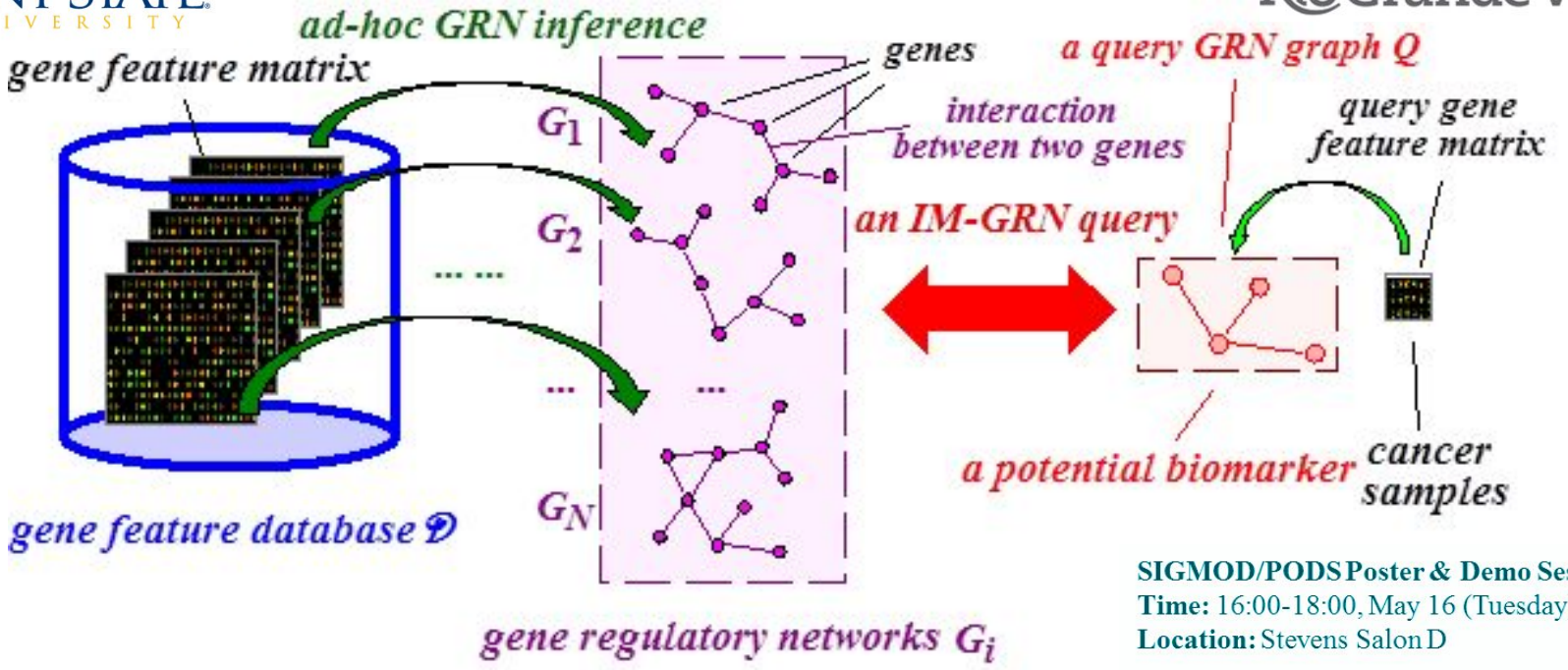
Session 8: Tree & Graph Processing 1, 14:00-15:40, Buckingham

Efficient Ad-Hoc Graph Inference and Matching in Biological Databases

Xiang Lian (Kent State U.) & D. Kim (U. of Texas Rio Grande Valley)

KENT STATE UNIVERSITY

The University of Texas Rio Grande Valley



SIGMOD/PODS Poster & Demo Session 1
Time: 16:00-18:00, May 16 (Tuesday)
Location: Stevens Salon D

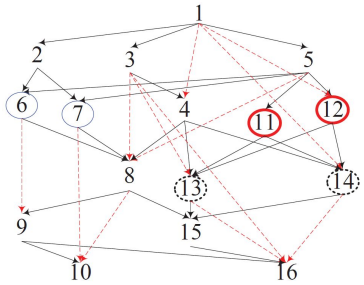
DAG Reduction: Fast Answering Reachability Queries

Tree & Graph

J. Zhou, S. Zhou. J. Xu Yu, H. Wei, Z. Chen, X. Tang

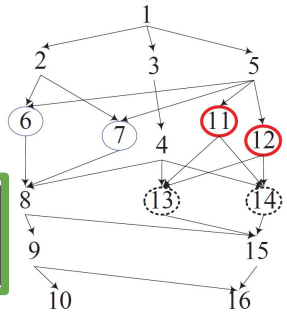
(Donghua U., Yanshan U., Chinese U. of Hong Kong, Shanghai Lixin U. of Commerce, Yanshan U.)

The Problem: Given a DAG G , find the *smallest* DAG by *TR* (transitive reduction) and *ER* (equivalence reduction)



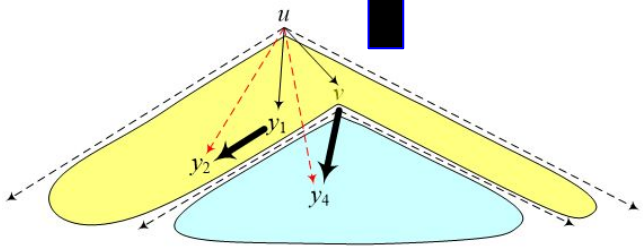
Remove redundant edge

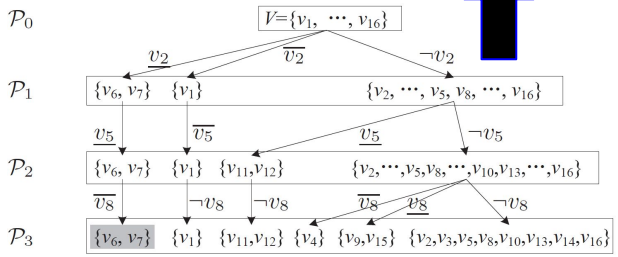
A *new* TR algorithm
- time complexity: $O(|V|+|E|+d\Delta|V|)$
- space complexity: $O(|V|)$

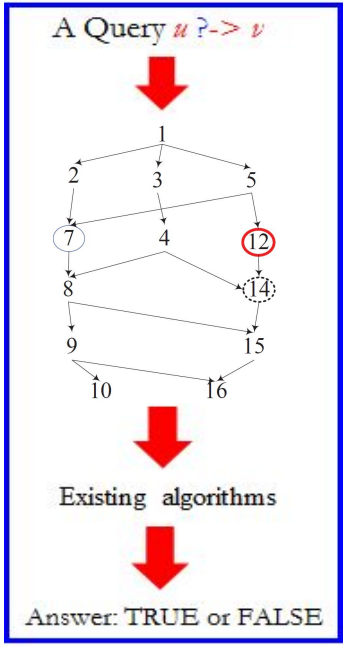


Remove equivalent nodes

A *linear* ER algorithm
- time complexity: $O(|V|+|E|)$
- space complexity: $O(|V|)$



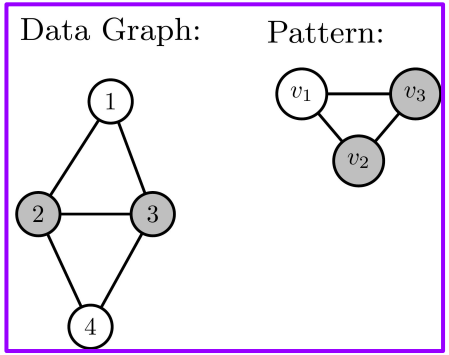




Flexible and Feasible Support Measures for Mining Frequent Patterns in Large Labeled Graphs

J. Meng & Y. Tu (U. of South Florida)

Problem: How to calculate support (frequency) of a pattern in a single graph?



Our hypergraph framework unifies two main supports and fills the gap between them

MNI support

Occurrences	v_1	v_2	v_3
f_1:	1	2	3
f_2:	1	3	2
f_3:	4	2	3
f_4:	4	3	2
# of images:	2	2	2

$\text{MNI} = \min(2,2,2) = 2$

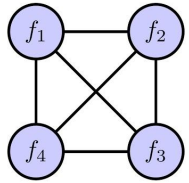
Linear time

Connections?

Minimum image based method

Overlap graph based method

MIS support



MIS = 1

NP-hard

New HW

Accelerating Pattern Matching Queries in Hybrid CPU-FPGA Architectures

David Sidler, Zsolt Istvan, Muhsen Owaida, Gustavo Alonso (ETH Zurich)



FPGAs are going to be in the processor!

Talk today

How to use FPGAs to accelerate string operations and text processing

Demo tomorrow

Hardware Operators:

- Regular Expression
- Skyline
- Stochastic Gradient Descent (SGD)

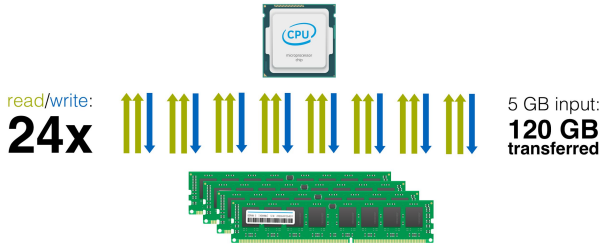
Free Swiss Chocolate!

A Memory Bandwidth-Efficient Hybrid Radix Sort on GPUs

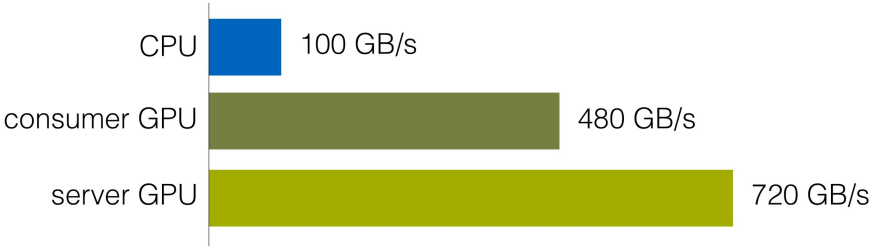
New HW

Elias Stehle & Hans-Arno Jacobsen (Technical University of Munich (TUM))

Sorting requires to read/write input many times



GPUs have extremely high memory bandwidth



We propose a faster GPU sorting algorithm:

baseline	theoretical speed-up	minimum speed-up	maximum speed-up
CUB, v. 1.5.1	*1.60x*	**1.6x**	**4.0x**
🔥 **CUB, v. 1.6.4**	*1.25x*	**1.2x**	**3.2x**
🔥 **Multisplit**	*1.25x*	**1.3x**	**1.8x**

🔥 most recent work

→ Up to **4.0x speed-up** over orig. baseline
→ **At least** 20% faster than most recent work

FPGA-based Data Partitioning

New HW

Kaan Kara, Jana Giceva, & Gustavo Alonso (ETH Zurich)

Partitioning data is necessary and expensive!

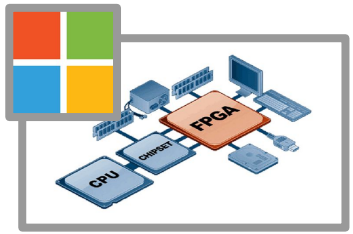
- Data sizes keep increasing while cache sizes do not.
- Partitioning has been fully optimized for a CPU.

Specialized hardware is becoming widely available.

Xeon+FPGA F1 instances Catapult





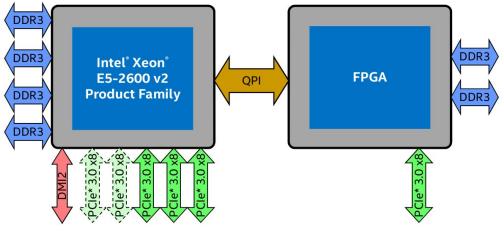


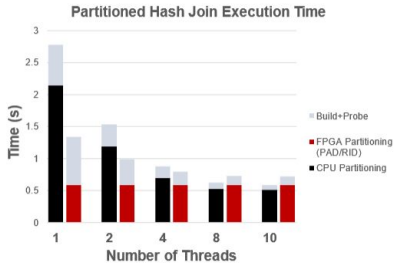
Can we perform data partitioning more efficiently using an FPGA?

Why would an FPGA be good at partitioning data?



How does a hybrid FPGA+CPU join work?



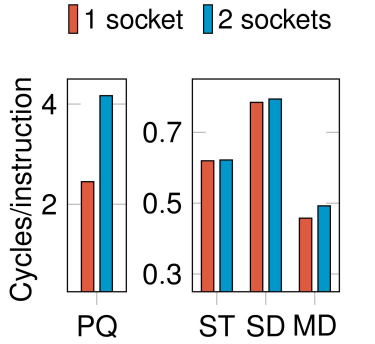


New HW

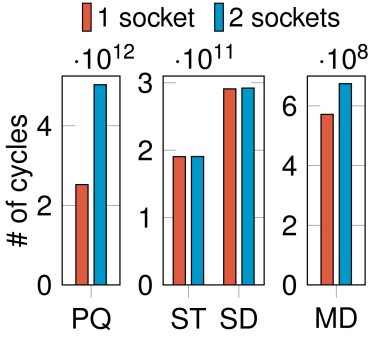
Template Skycube Algorithms for Heterogeneous Parallelism on Multicore and GPU Architectures

K. S. Bøgh, S. Chester, D. Šidlauskas, I. Assent (Aarhus U., NTNU, & EPFL)

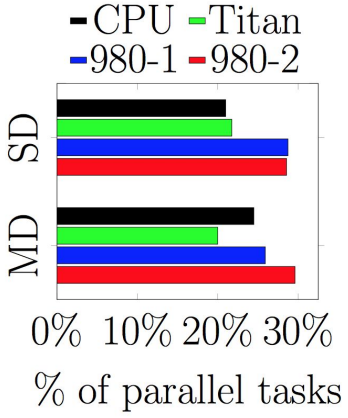
- Specialized accelerators, such as GPUs, are becoming omnipresent.
- How to meaningfully utilize all parts of a heterogenous system, for a single expensive operator, while retaining scalability and efficiency?
- **Redesign classic design pattern, for hardware specialization.**
- 4189x speedup in application on real data (8h -> 8 sec).

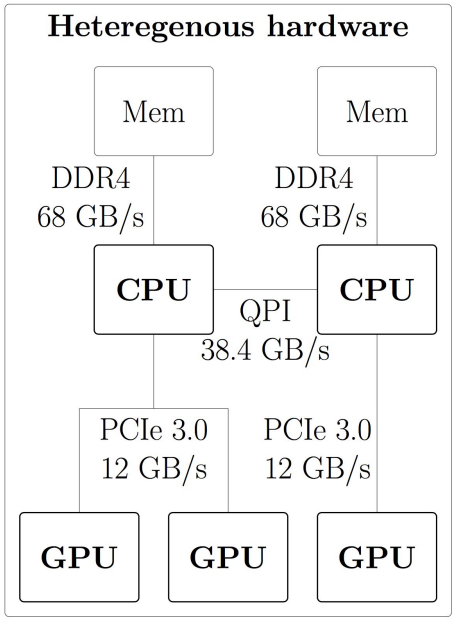


Cycles per instructions



Stalls, L3 pending loads





Heterogeneity-aware Distributed Parameter Servers

Parallel/Dist QP

Jiawei Jiang, Bin Cui, Ce Zhang, & Lele Yu (Peking Univ. & ETH Zurich)

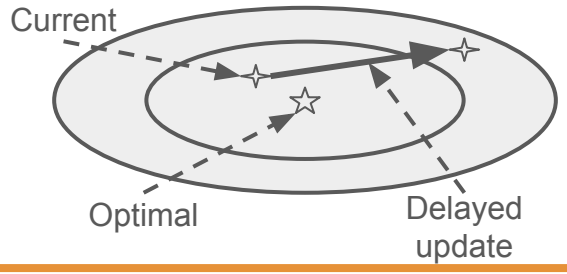
Problem: How to synchronize in distributed machine learning?

Current approaches

- BSP(Bulk Synchronous Parallel)
- ASP(Asynchronous Parallel)
- SSP(Stale Synchronous Parallel) ★

Challenge

- Heterogeneous cluster ⇨ Straggler
- Delayed update ⇨ Unstable SGD



Goal

- Per-worker global learning rate
- Consider the delay

Our approach

- Model the parameter server
- Model the staleness of updates
- Multi-version control

Result

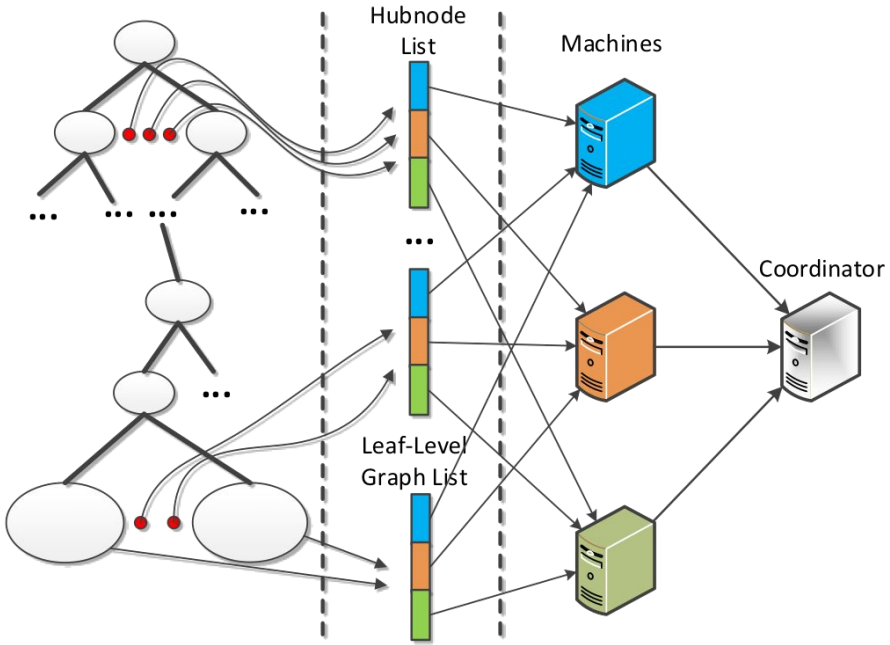
- Baselines: MLlib, Petuum, TensorFlow
- 2-12x speedup

Distributed Algorithms on Exact Personalized PageRank

Parallel/Dist QP

T. Guo, X. Cao, G. Cong, J. Lu, X. Lin (NTU, UNSW, & U. of Helsinki)

- Problem:
  - How to compute Personalized PageRank Vector in a distributed way?
- Challenges:
  - **Exactness**
  - **Parallel**
  - **Costs**. Time, space and network costs
- Solutions
  - Graph partitioning based algorithm
  - Hierarchical Graph partitioning
- Features
  - Exact
  - Load Balanced
  - Low communication cost
  - One time data transfer



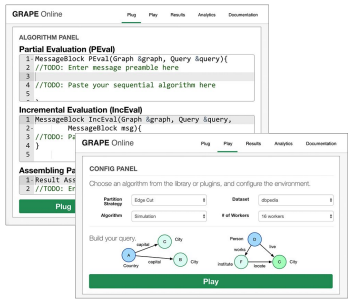
Parallelizing Sequential Graph Computations

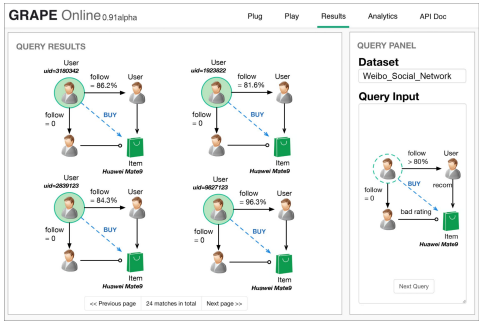
Parallel/Dist QP

W. Fan, J. Xu, Y. Wu, W. Yu, J. Jiang, Z. Zheng, B. Zhang, Y. Cao, C. Tian
(Univ. of Edinburgh, WSU, Hong Kong Baptist U., & Peking U.)

Motivation

- It is nontrivial for one to learn how to program in the new parallel models, e.g., "think like a vertex".
- Graph computations have been studied for decades, and a number of sequential graph algorithms are already in place. Can we use them without recasting?
- Do existing parallel graph engines guarantee termination and correctness?





GRAPE, parallelizing existing sequential algorithms as a whole, and guaranteeing *convergence* and *correctness* when the sequential algorithms provided are correct.

- Sequential graph algorithms can be *"plugged into"* GRAPE with minor additions, and get parallelized.
- *MapReduce*, *BSP* (bulk synchronous parallel) and *PRAM* models are *optimally simulated* by GRAPE.
- Foundation: a simuteneous fixed point computation with *partial evaluation* and *incremental evaluation*.

Features

- ***Ease of programming.*** Only need to provide three sequential algorithms for Q with minor additions.
- ***Semi-automated parallelization.*** Guarantee to converge at correct answers under a monotonic condition, if the three sequential algorithms provided are correct.
- ***Graph-level optimization.*** GRAPE inherits all optimization strategies available for sequential algorithms and graphs.
- ***Scale-up.*** GRAPE achieves comparable performance to the state-of-the-art graph systems.