Just In Time Datastructures

A Scalable Approach to Incremental Data Organization

Darshana Balakrishnan
dbalakri@buffalo.edu
SUNY Buffalo

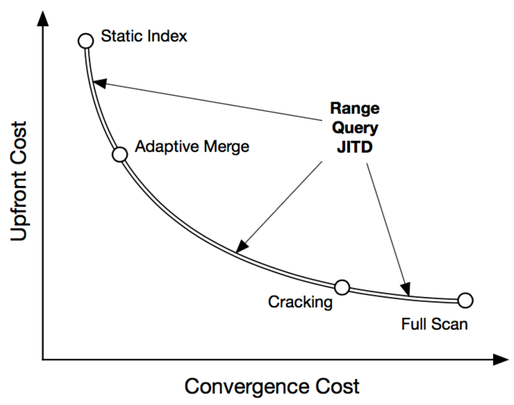
Saurav Singhi
sauravsi@buffalo.edu
SUNY Buffalo

Oliver Kennedy
okennedy@buffalo.edu
SUNY Buffalo

Lukaz Ziarek
lziarek@buffalo.edu
SUNY Buffalo

Background: Adaptive indexing

- Index update is a time consuming process as index structures need to be updated and reorganized every time the database changes.
- Adaptive indexing is a form of incremental index organization in which index creation happens as a side effect of query execution.
- **Advantage:** Incremental indexing puts a little penalty on the initial few queries but queries eventually get answered faster than a full static index evaluation.
- **Disadvantage:** An adaptive index structure like a sorted array or a B-Tree or adaptive indexing strategies like cracking, adaptive merge that is initially optimal for one workload becomes sub-optimal as the workload's characteristic changes.

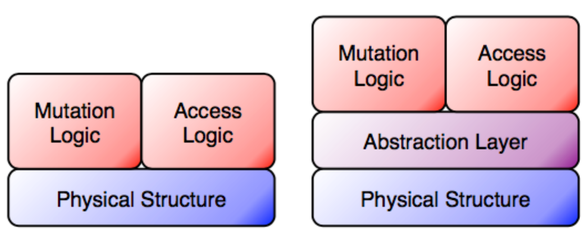


Just-In-Time Datastructures

Just-In-Time Data Structures is a generalized approach to adaptive indexing that dynamically adapts to changing workloads even after index convergence. It is possible to simulate the behavior of the Just-In-Time data Structure so as to predict appropriate data structure and heuristics for index creation also making the structure optimal for any change in workloads.

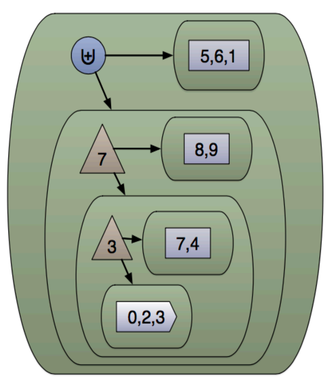
Just-in-time data structures:

- Driven by abstraction between the physical representation and logic that accesses and manipulates that physical representation.
- **Cogs(Abstraction layer)**: Collection of nodes in the data structure irrespective of their type. JITDs composed of interchangeable cogs that are consistent in the data they store but need not be consistent in the structure of the nodes.
- **Policies**: In place transformations that restructure the JITD are interleaved with the actual query execution making the JITD an iterative and incremental model rather than a static indexed structure.

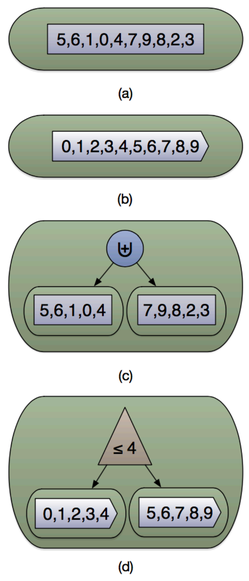


(a) A Traditional Datastructure's Design

(b) A Just-In-Time Datastructure's Design



Example of JITD



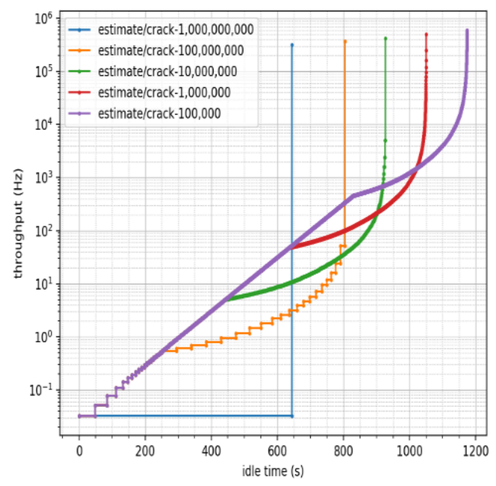
Different types of cogs

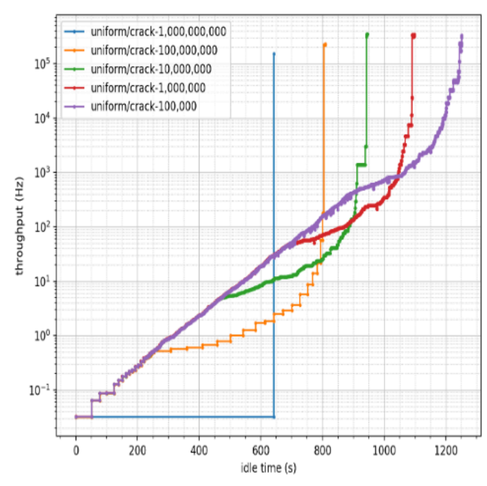
Simulator Model

- **Motivation**: Execution of the JITDs on a workload to predict the best data structure and transformations that maximize the throughput and other relevant value functions is a very expensive process .
- **Simulation using cost Model**: For the purpose of estimation we defined a cost model that relates the different factors that contribute to the latency of the system. Each policy was broken down into a set of operations and cost of each operation was mathematically realized.
- Individual Costs measured for the following operations:

Operation on an array of size n	Time taken(ns)
Scan Unsorted Array (μ)	31.03
Scan Sorted Array (α)	104.59
Crack an Unsorted Array (ϑ)	48.42
Crack an Unsorted Array (Ω)	21.55

- Using the measured values we can calculate time taken for each operation. Total Time taken by the JITD is the sum of the time taken by the different operations dictated by the policy in place
  - Calculated time to Crack an Unsorted Array of size n = $\vartheta * n$.
  - Calculated time to Sort an Unsorted Array of size n = $\Omega * nlog(n)$.
  - Calculated time to Scan Unsorted Array of size n = $\mu * n$.
  - Calculated time to Scan Sorted Array of size n = $\alpha * log(n)$.





Conclusions

- JITDs decouple the logic and physical representation of an index data structure, and allow multiple behaviors, or policies to collectively manipulate a standardized library of physical layout building blocks, or cogs.
- The simulator model predicts the behavior of the JITD with a very minimal difference between the projected runtimes and calculated runtimes.
- The simulator model will help predict the intermediate state of a datastructure in transition.
- Simulation + Cost-Analysis can be used to derive policies to drive direct rewrites.

References

1. S. Idreos, M. L. Kersten, and S. Manegold. Database cracking. In CIDR, 2007.
2. O.kennedy, L.Ziarek. Just In Time Datastructures. In CIDR, 2015.