Kairo – Incremental View Maintenance For Scalable Virtual Switch Caching

Annus Zulfiqar, Ben Pfaff[1], Gianni Antichi[2], Muhammad Shahbaz

University of Michigan, [1]Feldera, [2]Politecnico di Milano

Virtual Switches Experience Frequent Rule Updates

- Virtual switches (vSwitches) optimize performance by caching **multi-table lookups into single-table caches** and **ensure consistency by revalidating the entire cache** every second
- The **operational environments** often require **frequent rule churn** arising from policy updates, periodic maintenance, service chain updates, load balancing, auto-scaling services, security responses, and flow expiry

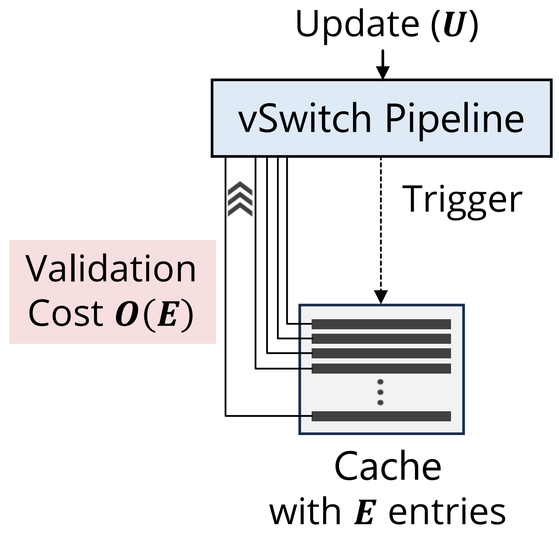
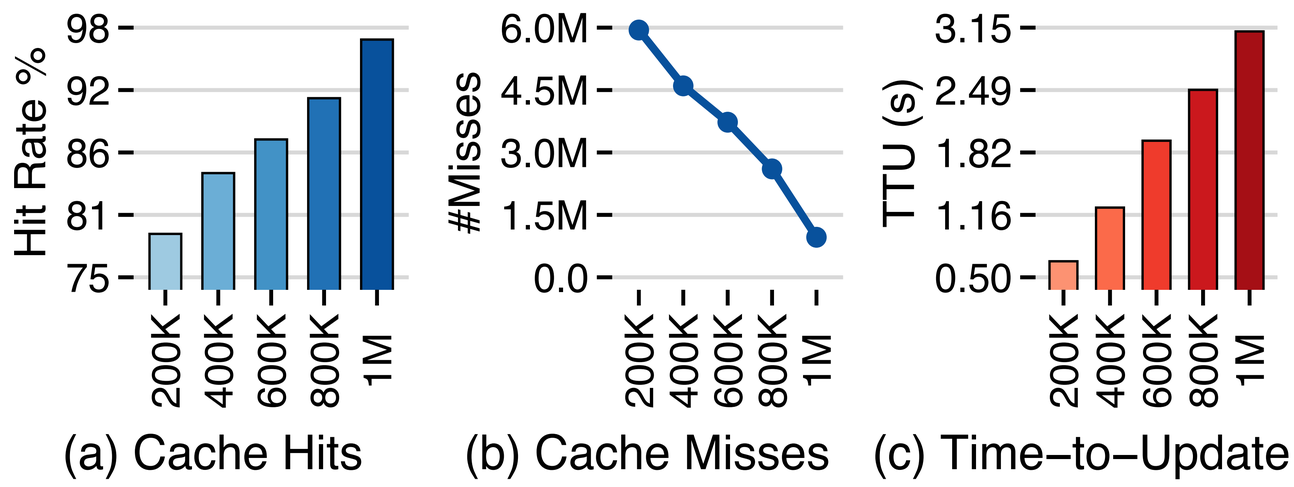


Fig 1: Traditionally, vSwitch rule updates require bottom-up full cache revalidation

Rule Updates are a Major Performance Bottleneck in vSwitches

- Scaling to **larger cache size significantly improves vSwitch performance** owing to higher hit rates and lower cache misses
- But the **cost of updating the vSwitch** also scales **proportional to cache size**
- To support vSwitch updates in a reasonable time interval (1 sec), ***OVS limits the cache size to only 200K!***



(a) Cache Hits (b) Cache Misses (c) Time–to–Update

Fig 2: vSwitch performance scales with cache size but supporting rule updates in a reasonable time severely limits the realizable benefits

Towards Incremental View Maintenance for vSwitch Updates

- Kairo frames **vSwitch updates** as an instance of the **Incremental View Maintenance (IVM)** problem and supports updates by **reacting only to rule changes in a top-down manner**
- Kairo maintains **traversals—linear, unrolled paths through the vSwitch pipeline**—as first-class **queries that capture the decision logic** for each individual **control flow of the rule set**
- As rule updates (ΔR) are much smaller than the cache size (E), an IVM engine such as DBSP can update a **200K entry cache in 3.2ms vs 670ms** for traditional bottom-up updates in OVS!

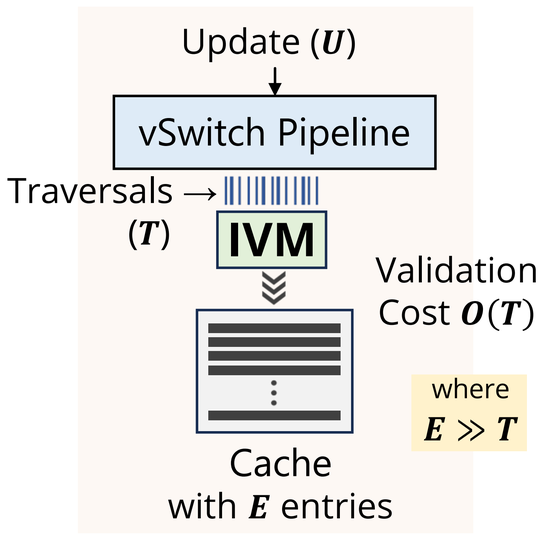


Fig 3: IVM has the potential for efficient top-down updates

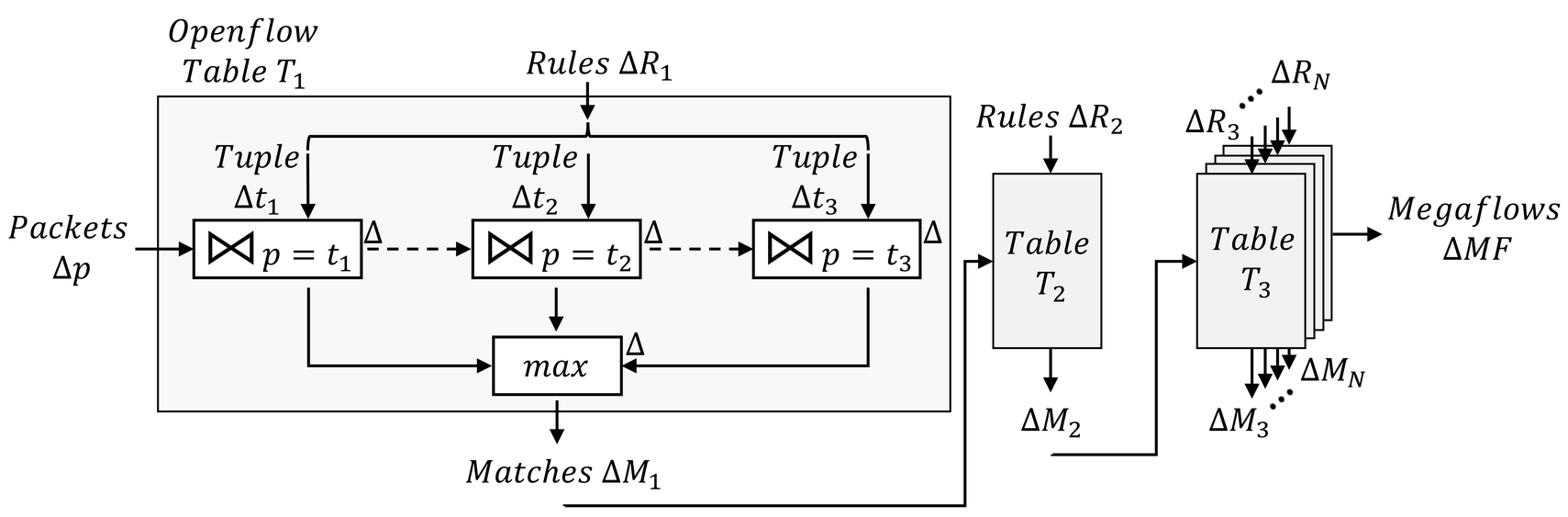


Fig 4: A DBSP circuit representation of an Openflow pipeline with rules and packets as input streams and IVM for efficient cache updates

Contact us: zulfiqaa@umich.edu