MVEDSUA: Higher Availability Dynamic Software Updates via Multi-Version Execution

Luís Pina
George Mason University
lpina2@gmu.edu

Anastasios Andronidis
Imperial College London
a.andronat15@imperial.ac.uk

Michael Hicks
University of Maryland
mwh@cs.umd.edu

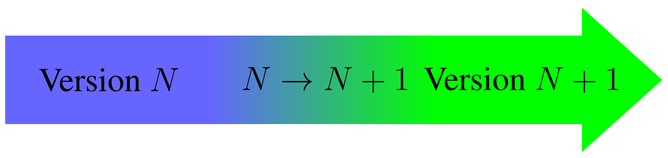
Cristian Cadar
Imperial College London
c.cadar@imperial.ac.uk

Software Updates are Unavoidable . . .

- Deploy new features, improve performance, fix bugs
- Remove known vulnerabilities

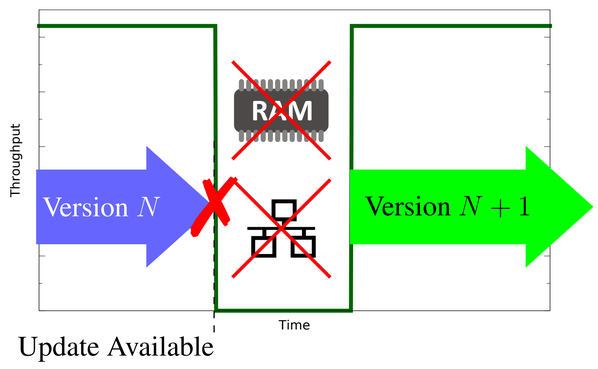
- E.g., Equifax used a vulnerable Struts version for more than 2 months, resulting the known compromise of sensitive information

Dynamic Software Updating Helps . . .



- Update in process, migrate state during small update pause
- Requires developer support: Updates as any other feature

. . . But Disruptive (at best)



- Typical update: Stop old process, restart in new version
- Loss of state: Contents of memory, active network connections, kernel state
- Period of no service

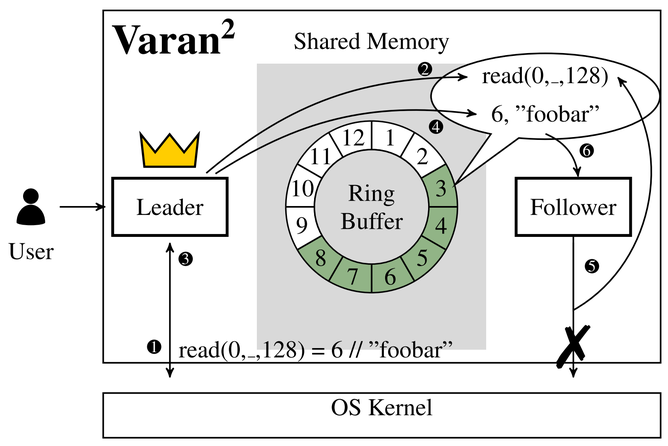
. . . But Suffers From Failed Updates

- Errors in updates can still crash, hang, corrupt data, or lose data
- How to recover from crashes/hangs? How to detect corrupt/lost data?

Rolling Upgrades and State

- Updating one node at the time drops node state
- Facebook uses custom Memcached for caches to avoid losing state[1]

Multi-Version Execution (MVE)



Leader:

1. Intercept `read` syscall
2. Save (1) to shared mem
3. Intercept syscall return
4. Save (2) and data to shared mem

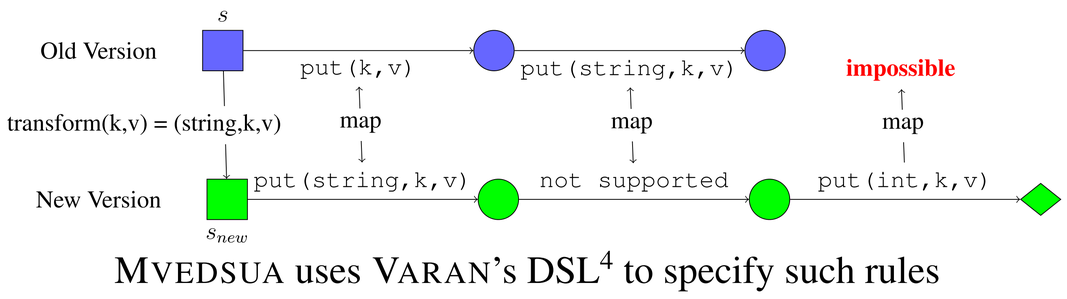
Follower:

5. Intercept syscall and match with shared mem
6. Get result/data from shared mem

Matching Semantics

Example:

- Key/value store with wire protocol: `put(k,v)`
- Updated to `put(type,k,v)` with `type={string,int}`
- Update turns all pairs `(k,v)` to `(string,k,v)`
- Mapping function needed to match semantics between versions:



MVEDSUA uses VARAN's DSL[4] to specify such rules

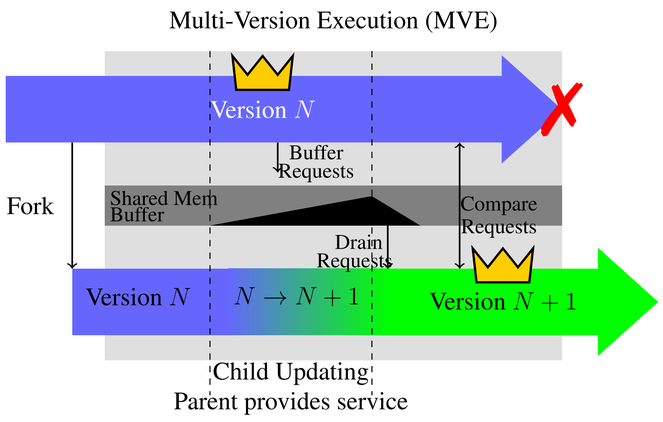
[1] Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C. Li, Ryan McElroy, Mike Paleczny, Daniel Peek, Paul Saab, David Stafford, Tony Tung, and Venkateshwaran Venkataramani. *Scaling Memcache at Facebook*

[2] Petr Hosek and Cristian Cadar. *Varan the Unbelievable: An Efficient N-version Execution Framework*

[3] Christopher M. Hayden, Edward K. Smith, Michail Denchev, Michael Hicks, and Jeffrey S. Foster. *Kitsune: Efficient, General-purpose Dynamic Software Updating for C.*

[4] Luís Pina, Daniel Grumberg, Anastasios Andronidis, and Cristian Cadar. *A DSL Approach to Reconcile Equivalent Divergent Program Executions*

MVEDSUA: Combine DSU with MVE

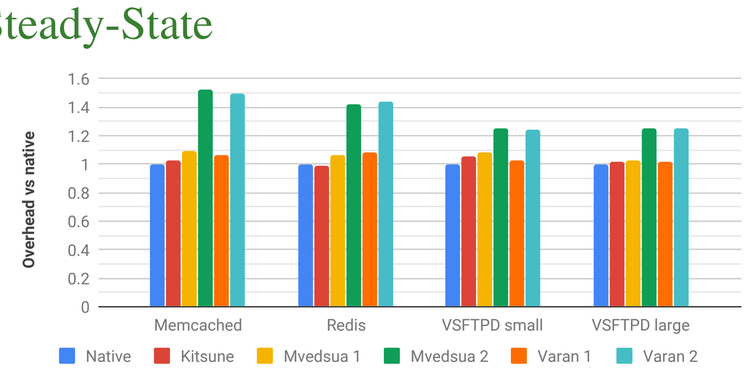


- Fork when update available
- Start MVE with parent as leader and child as follower
- Update follower while leader provides service
- Compare requests to detect state corruption
- Error in update: Terminate follower
- Successful update: Promote new version and terminate old version

MVEDSUA uses Kitsune[3] for DSU and VARAN[2] for MVE

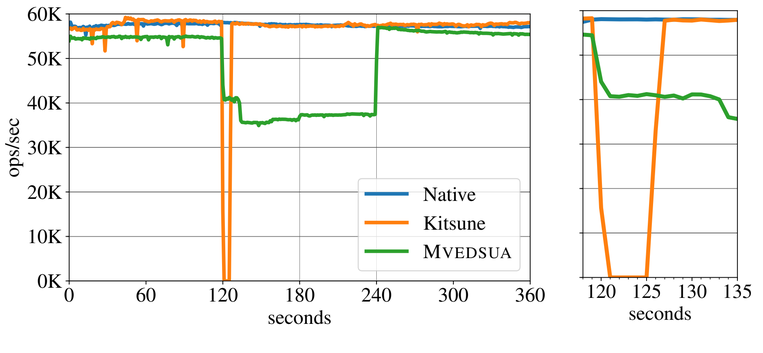
Performance

Steady-State



- 1-version MVEDSUA imposes low overhead
- Majority of time spent in 1-version
- 2-version overhead comes from VARAN

Update



- Updating Redis with 1M entries, totaling 250MB of memory
- Kitsune imposes 5 second pause
- MVEDSUA eliminates pause (and tolerates errors)