etcd: mission critical key-value store

Gopherfest
16 May 2016

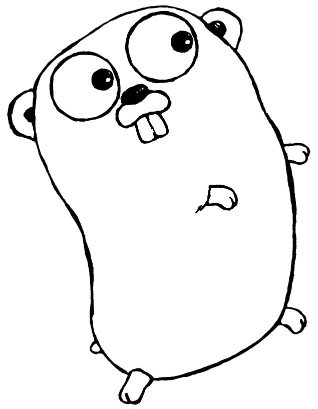
Gyu-Ho Lee
CoreOS

Welcome

Slides are here:

github.com/gyuho/presentations (https://github.com/gyuho/presentations)

Go + etcd



Agenda

- What is etcd
- Why
- Go
- Q/A

What is etcd

etcd is ...

- Distributed key-value store
- Open source github.com/coreos/etcd (https://github.com/coreos/etcd) (~ June 2013)
- Still new, compared to ZooKeeper (~ May 2008)
- Google Kubernetes (http://kubernetes.io/), YouTube Doorman (https://github.com/youtube/doorman), ...
- Red Hat, EMC, Cisco, Huawei, Baidu, Alibaba...

etcd API

```
cli.Put(ctx, "foo", "bar", Lease)
cli.Get(ctx, "foo")
cli.Delete(ctx, "foo")

// Transaction
kvc.Txn(ctx).
 If(clientv3.Compare(clientv3.Value("key"), ">", "abc")). // txn value comparisons are lexical
 Then(clientv3.OpPut("key", "XYZ")).                      // this runs, since 'xyz' > 'abc'
 Else(clientv3.OpPut("key", "ABC")).
 Commit()

// Watch for updates on key
ch := cli.Watch(ctx, "foo")
for res := range ch {}

// Distributed locks
mu := concurrency.NewMutex(cli, "foo")
mu.Lock()
mu.Unlock()
```

Why etcd

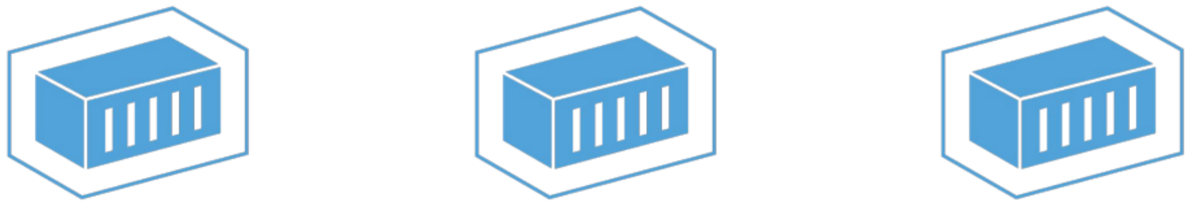
Use etcd to store configuration

For small chunks of data

```
maxReqBytes = 1.5 * 1024 * 1024  // 1.5MB

DefaultQuotaBytes = int64(2 * 1024 * 1024 * 1024) // 2GB
MaxQuotaBytes = int64(8 * 1024 * 1024 * 1024)     // 8GB
```

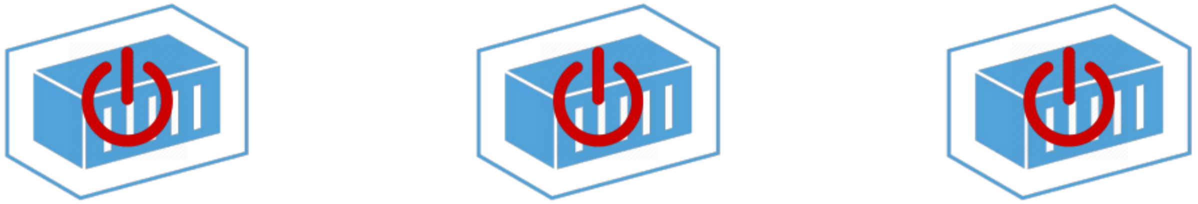
Updates

Security updates?



How would you update the cluster of machines?

Traditional way

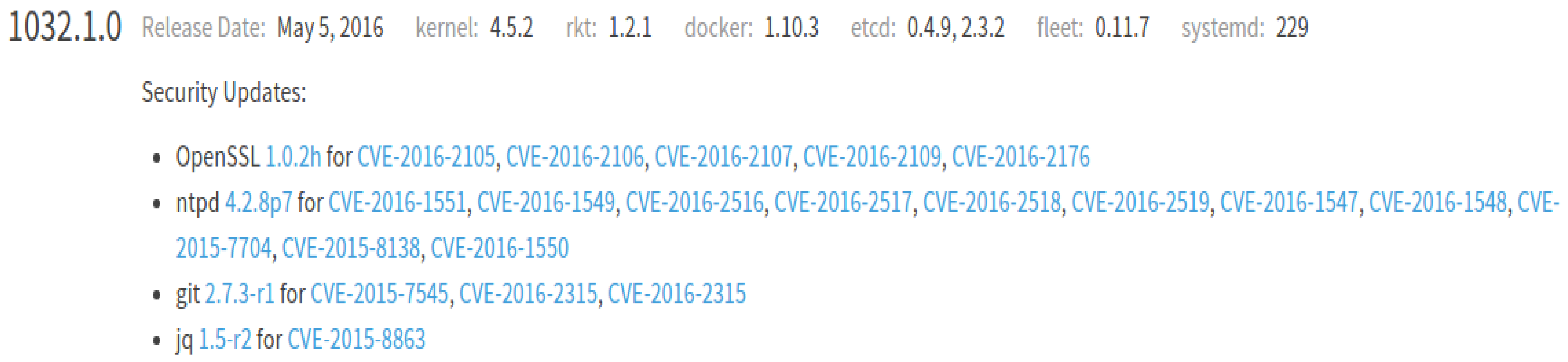


- Reboot with downtime
- Too Manual

CoreOS updates with etcd

If you run your application on CoreOS,

your OS gets Automatic, No-downtime updates



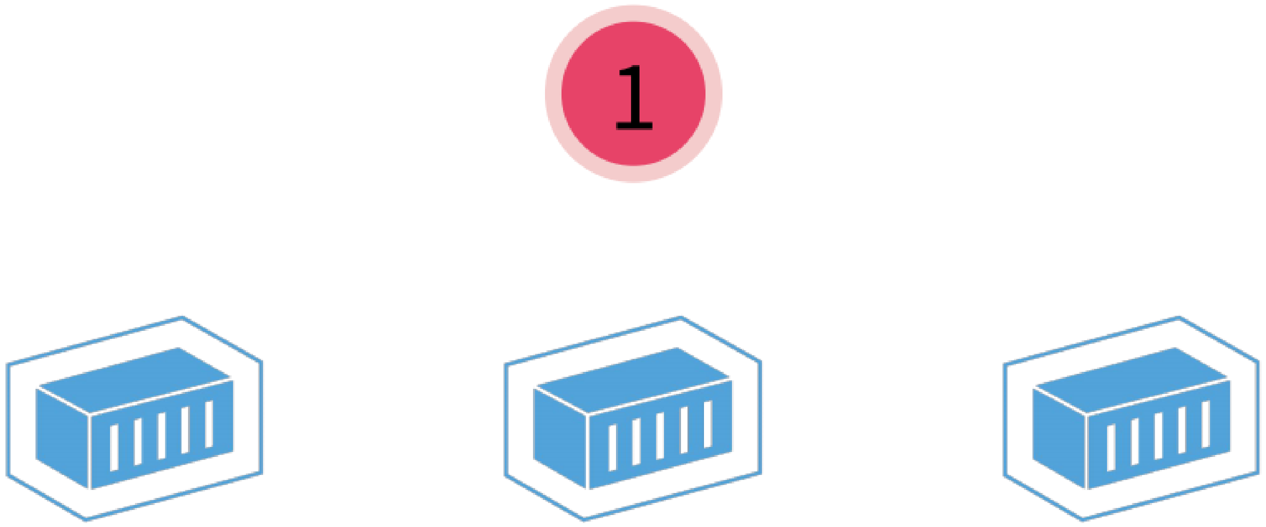
1032.1.0 Release Date: May 5, 2016 kernel: 4.5.2 rkt: 1.2.1 docker: 1.10.3 etcd: 0.4.9, 2.3.2 fleet: 0.11.7 systemd: 229

Security Updates:

- OpenSSL 1.0.2h for CVE-2016-2105, CVE-2016-2106, CVE-2016-2107, CVE-2016-2109, CVE-2016-2176
- ntpd 4.2.8p7 for CVE-2016-1551, CVE-2016-1549, CVE-2016-2516, CVE-2016-2517, CVE-2016-2518, CVE-2016-2519, CVE-2016-1547, CVE-2016-1548, CVE-2015-7704, CVE-2015-8138, CVE-2016-1550
- git 2.7.3-r1 for CVE-2015-7545, CVE-2016-2315, CVE-2016-2315
- jq 1.5-r2 for CVE-2015-8863

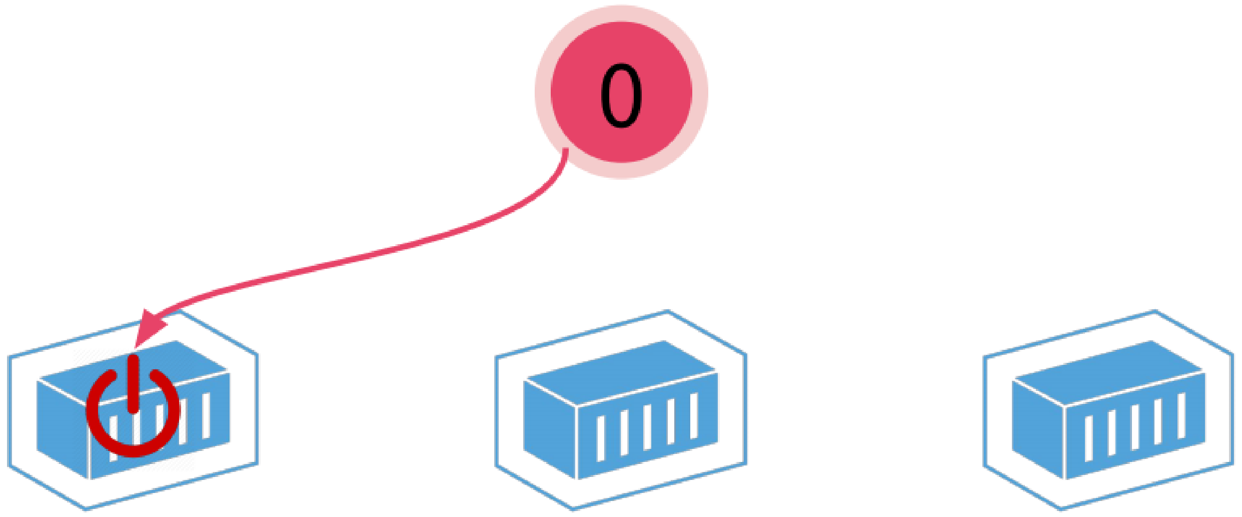
CoreOS updates with etcd

- CoreOS updates are done by locksmith (https://github.com/coreos/locksmith)
- locksmith is built on top of etcd
- locksmith stores semephore values in etcd
- ensure that only subset of cluster are rebooting at any given time



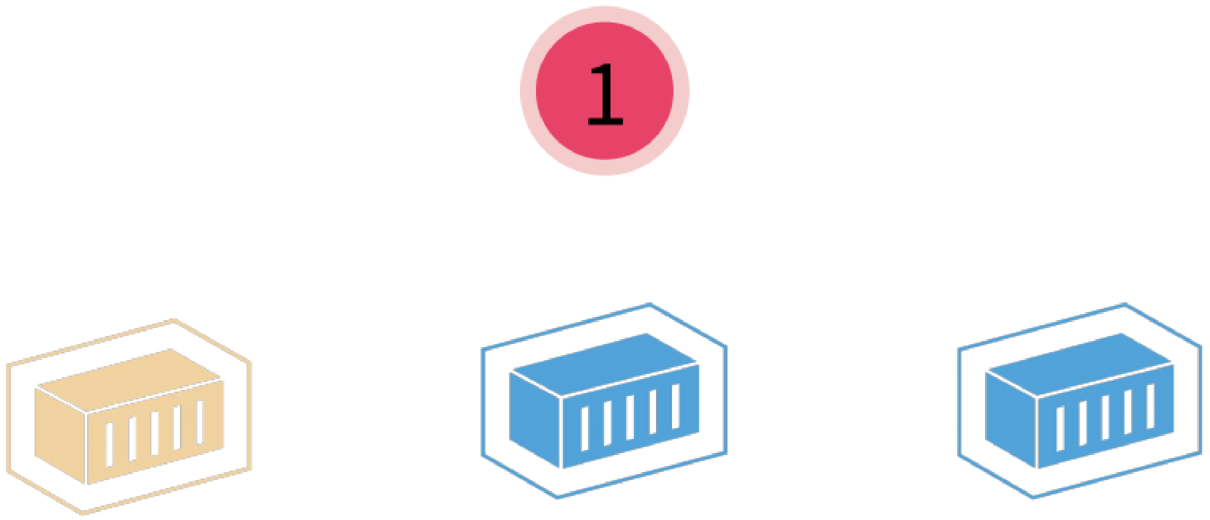
CoreOS updates with etcd

Decrement semaphore when rebooting



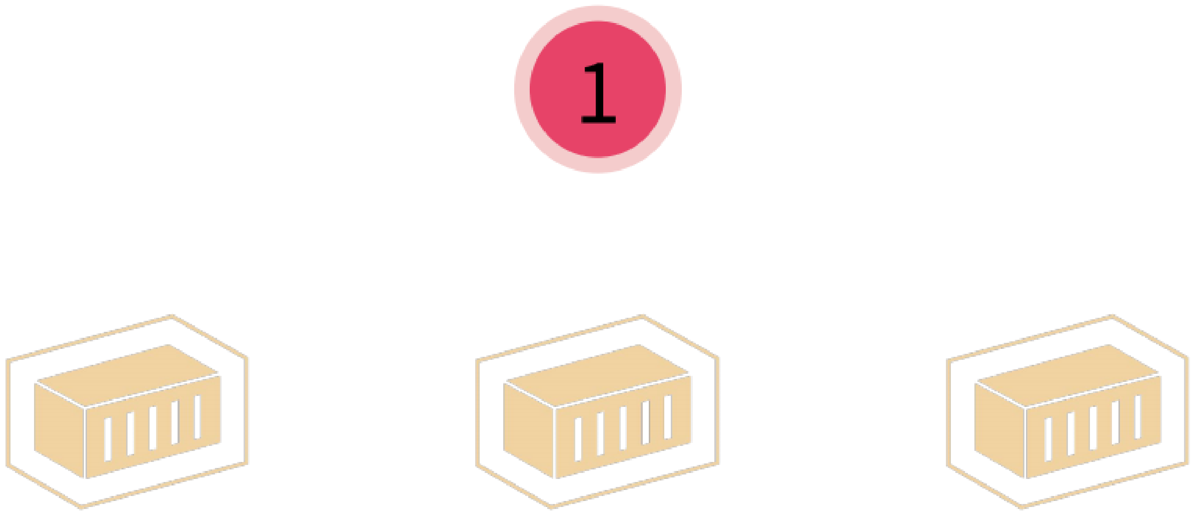
CoreOS updates with etcd

- Automatic
- No downtime



CoreOS updates with etcd

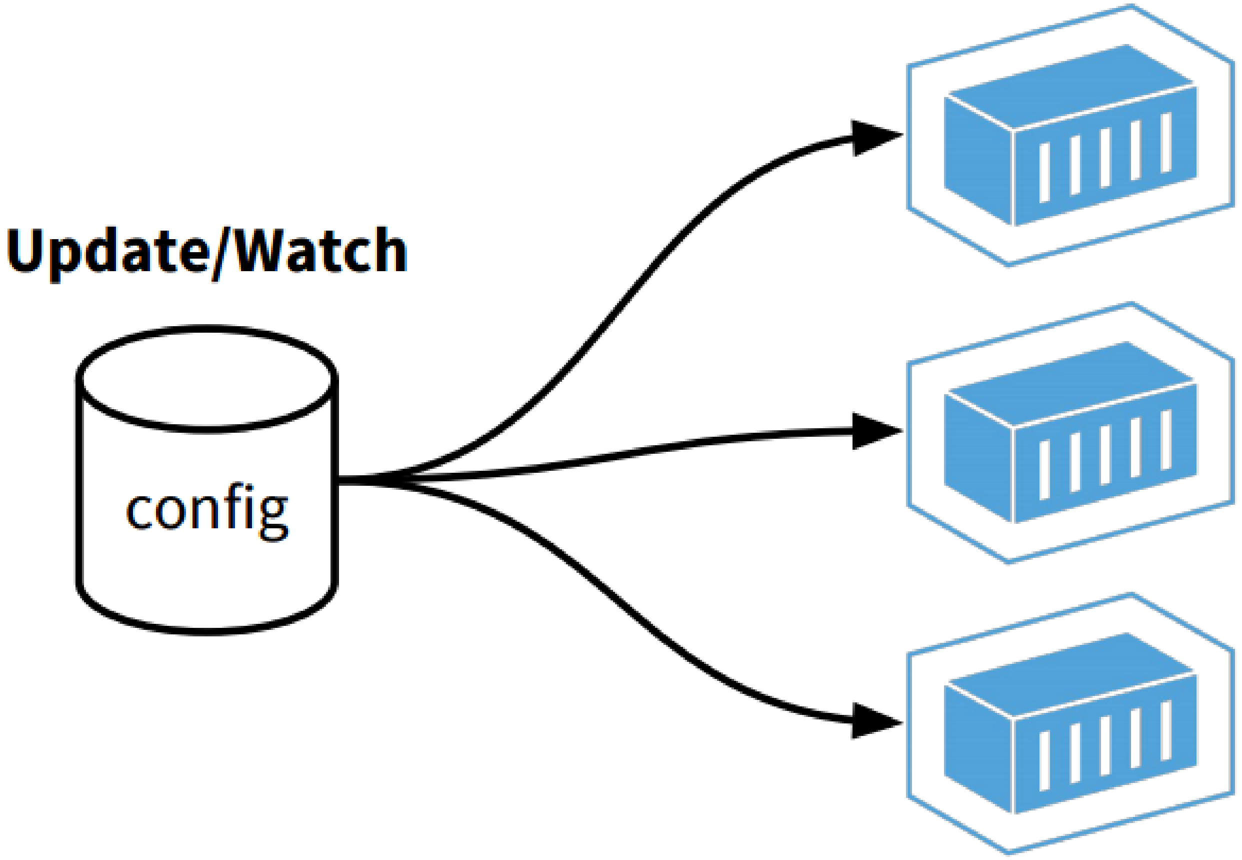
Your cluster is now secured



Use etcd for "critical" configuration

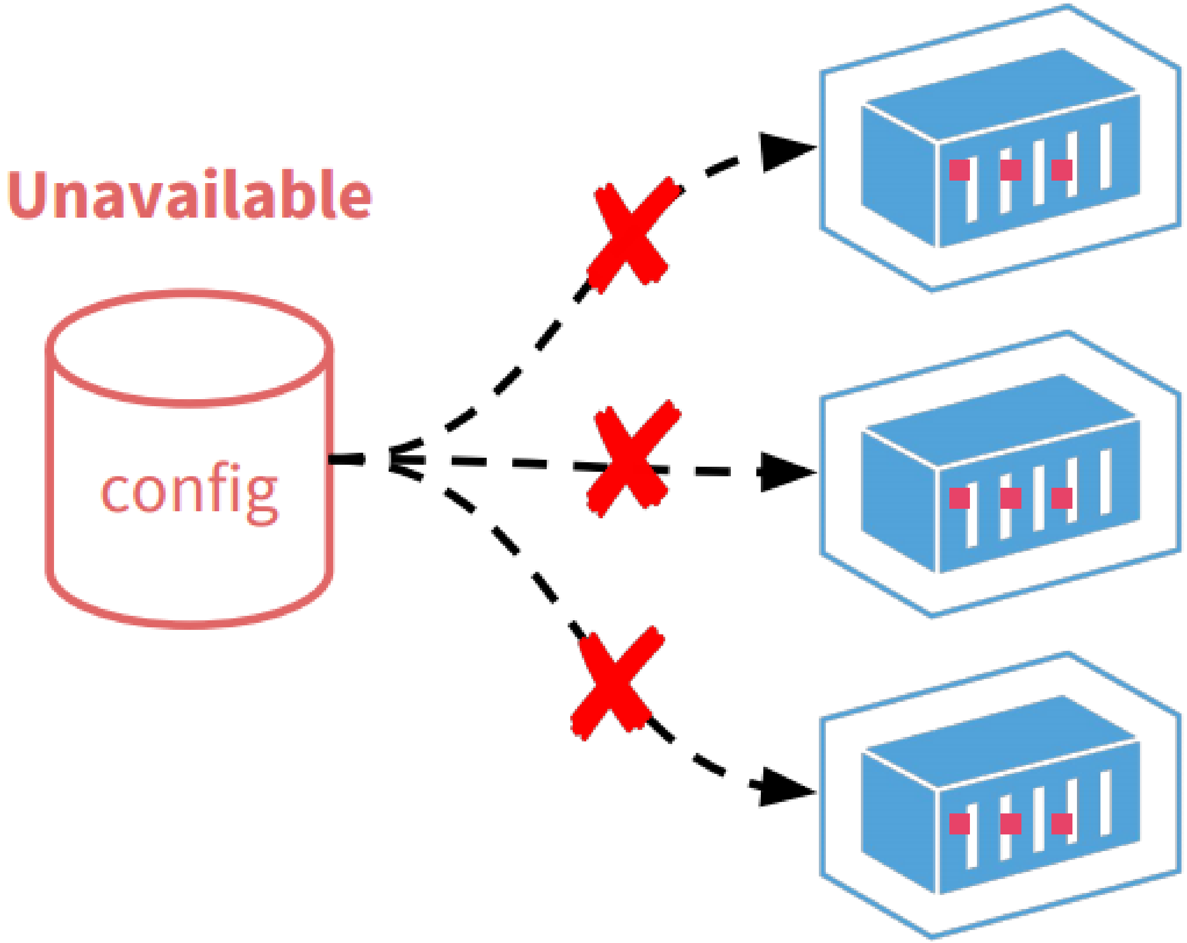
Bad practice

What if this machine goes down?



Bad practice

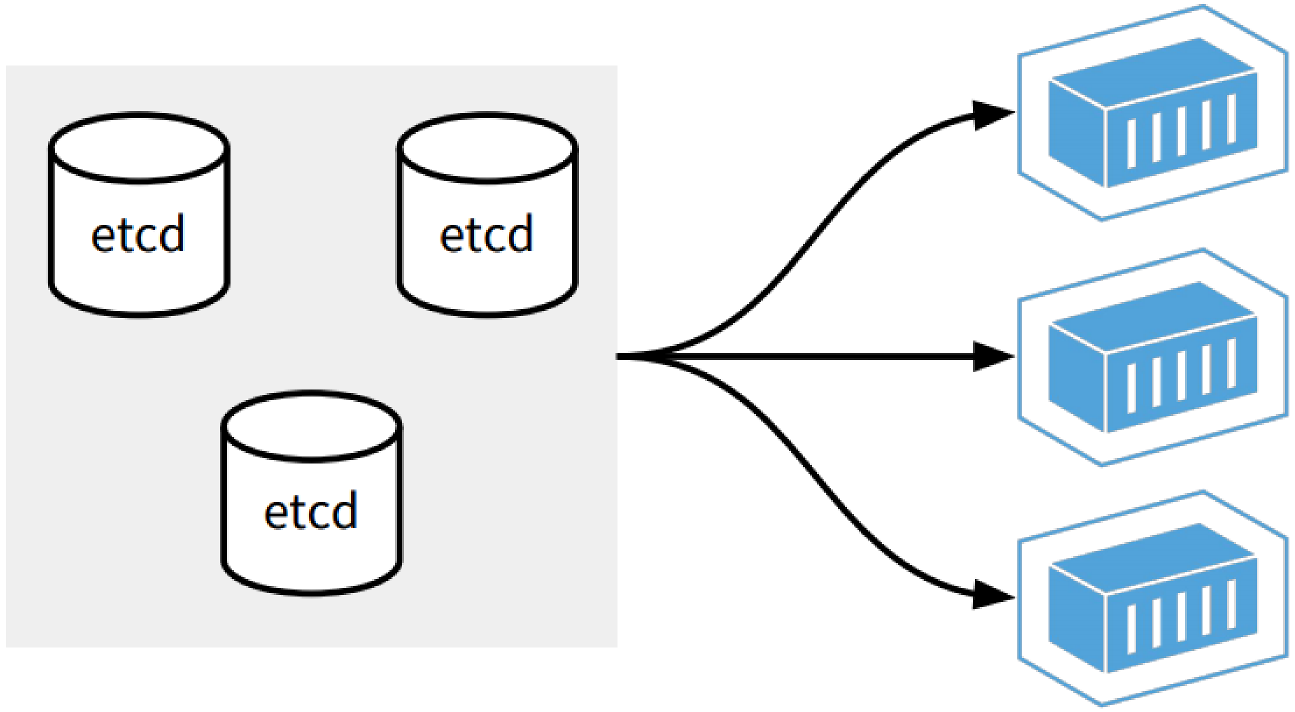
Single point of failure



Good practice

etcd replicate your data to multiple machine

and still provides consistent view of your data

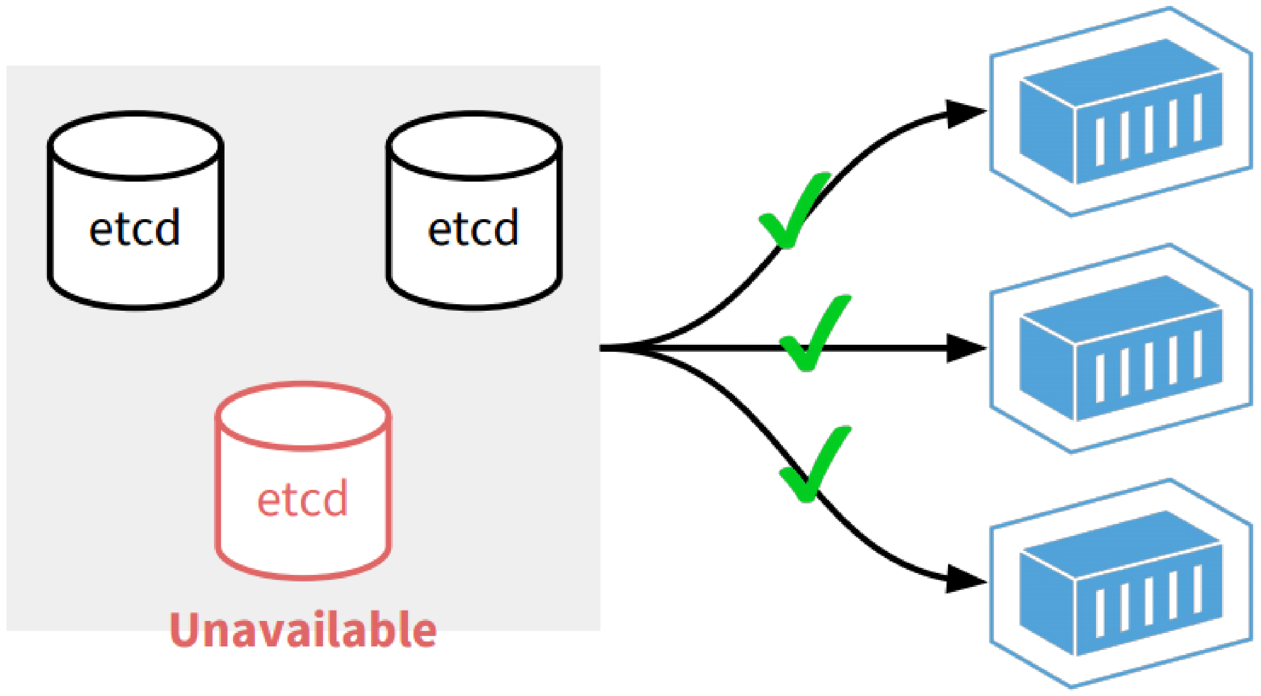


Good practice

etcd can tolerate machine failures

Can tolerate up to 1 out of 3-node cluster

Can tolerate up to 2 out of 5-node cluster



Demo

play.etcd.io (http://play.etcd.io)

Join me!

etcd

Consistent view of critical configuration

- Strong consistency (no stale reads)
- Different than eventual consistency (conflicts, latest timestamps wins)

Highly available configuration store

- Resilient to a single point of failures & network partitions

Watchable

- Push configuration updates to application

Why not ZooKeeper or Consul?

They are all great projects.

They have their own use cases.

etcd is built for scalability and reliability.

etcd Project Status: Performance

etcd v3

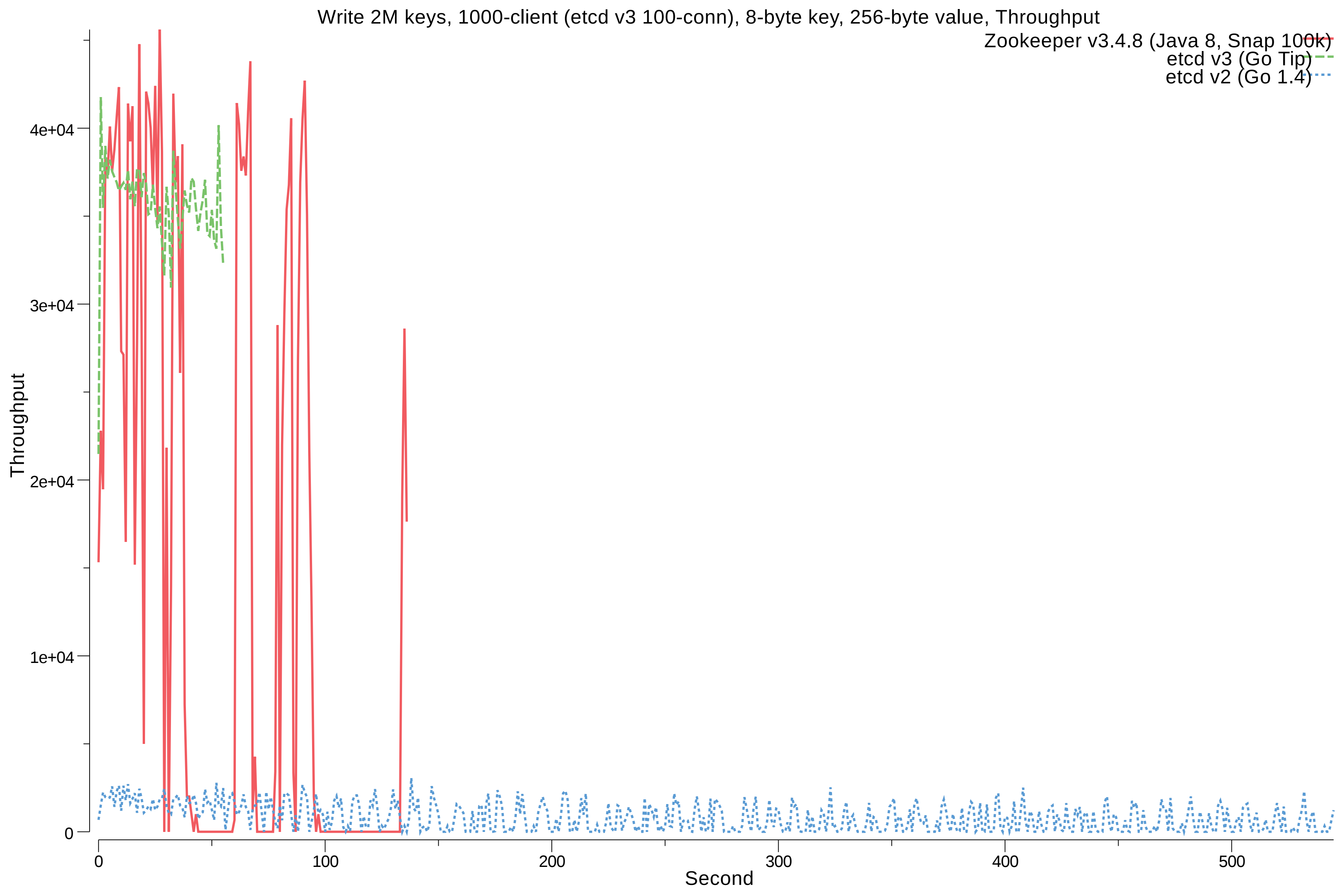
BoltDB (https://github.com/boltdb/bolt)

- B+tree disk storage
- Incremental snapshot
- vs. ZooKeeper snapCount 10,000

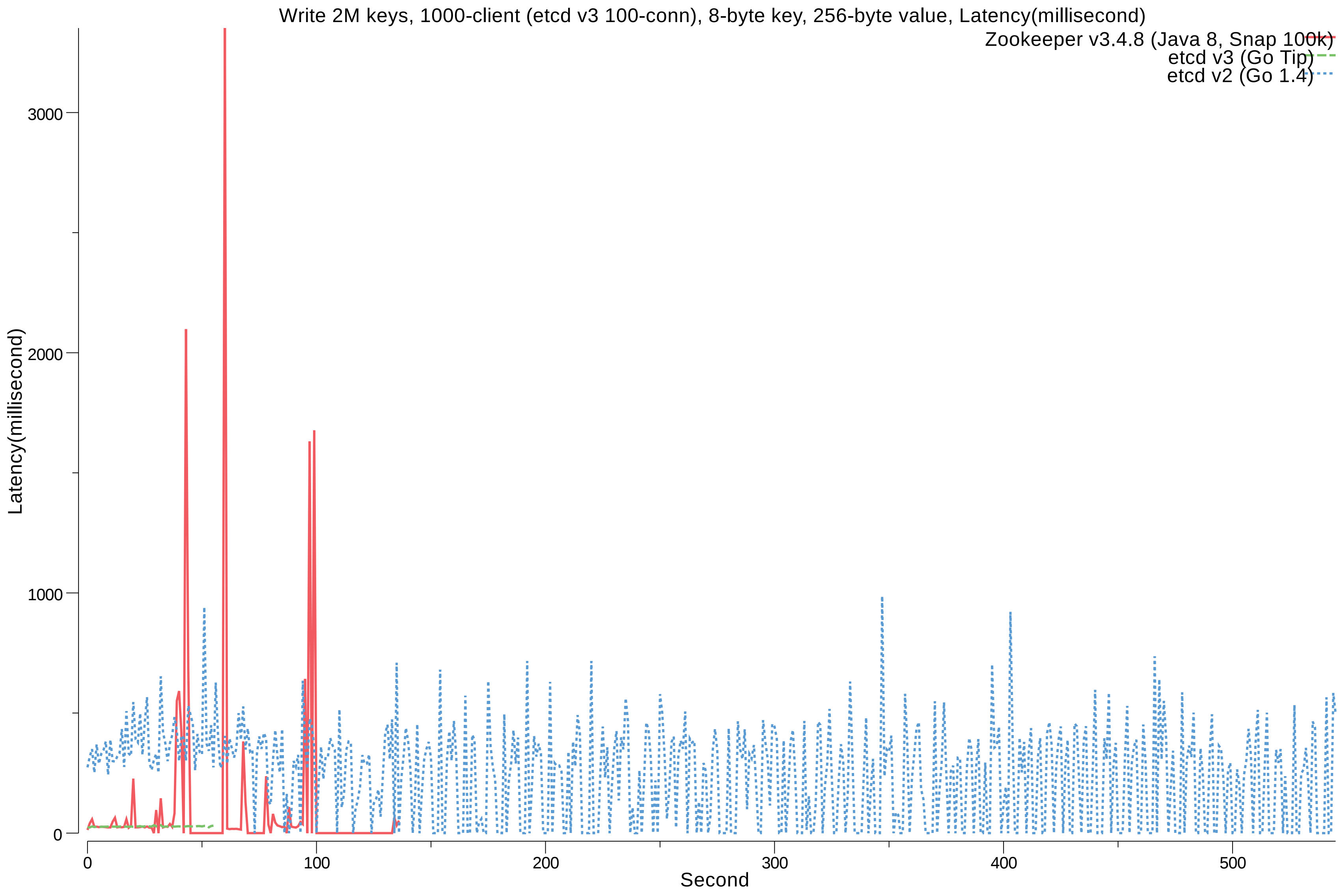
gRPC (http://www.grpc.io/)

- Protocol Buffer
- HTTP/2
- streams, less TCP congestions

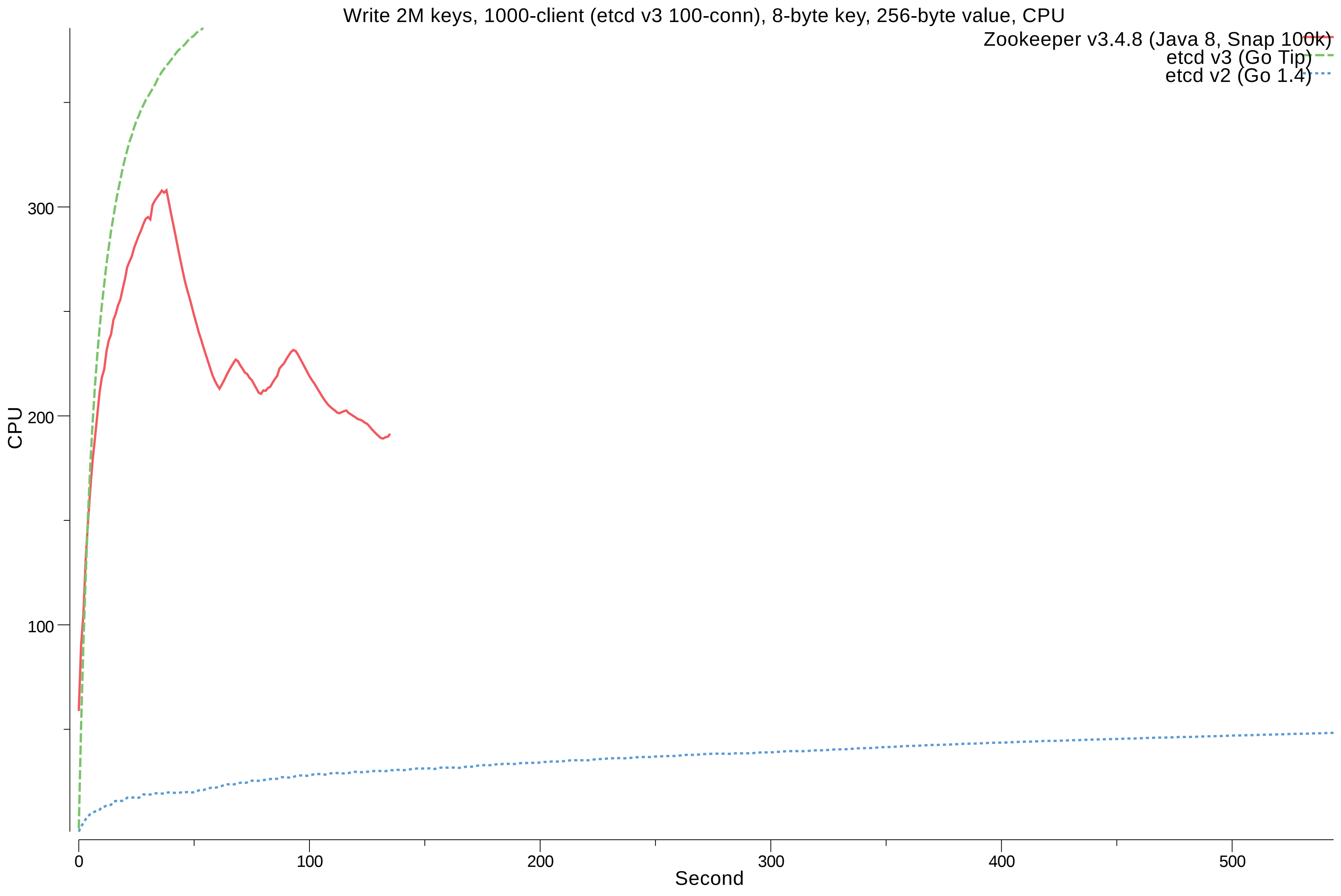
Throughput


Write 2M keys, 1000-client (etcd v3 100-conn), 8-byte key, 256-byte value, Throughput
Zookeeper v3.4.8 (Java 8, Snap 100k)
etcd v3 (Go Tip)
etcd v2 (Go 1.4)
Throughput
0
1e+04
2e+04
3e+04
4e+04
Second
0
100
200
300
400
500

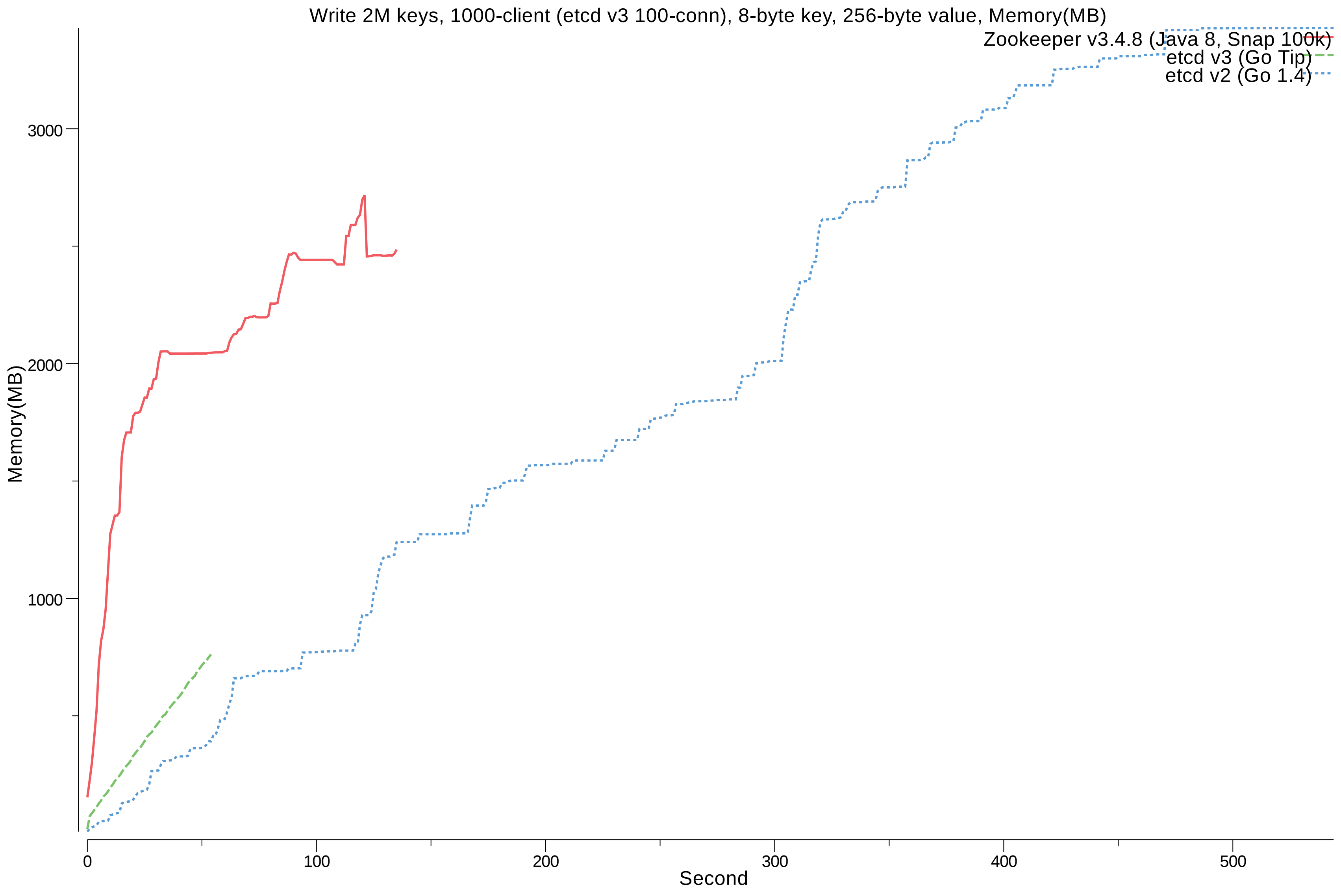
Latency


Write 2M keys, 1000-client (etcd v3 100-conn), 8-byte key, 256-byte value, Latency(millisecond)
Zookeeper v3.4.8 (Java 8, Snap 100k)
etcd v3 (Go Tip)
etcd v2 (Go 1.4)
Latency(millisecond)
3000
2000
1000
0
0
100
200
300
400
500
Second

CPU

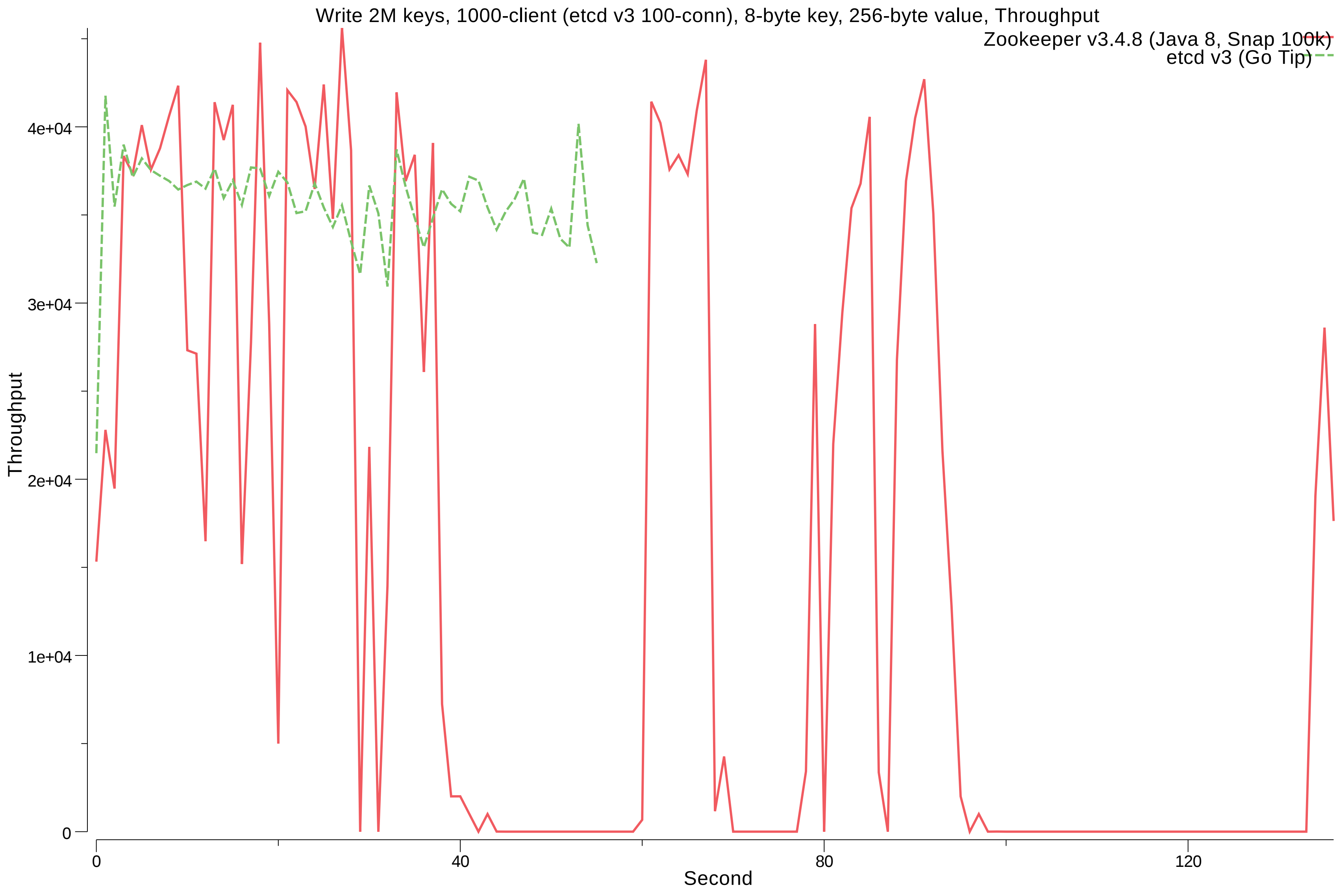

Write 2M keys, 1000-client (etcd v3 100-conn), 8-byte key, 256-byte value, CPU
Zookeeper v3.4.8 (Java 8, Snap 100k)
etcd v3 (Go Tip)
etcd v2 (Go 1.4)
CPU
300
200
100
0
100
200
300
400
500
Second

Memory

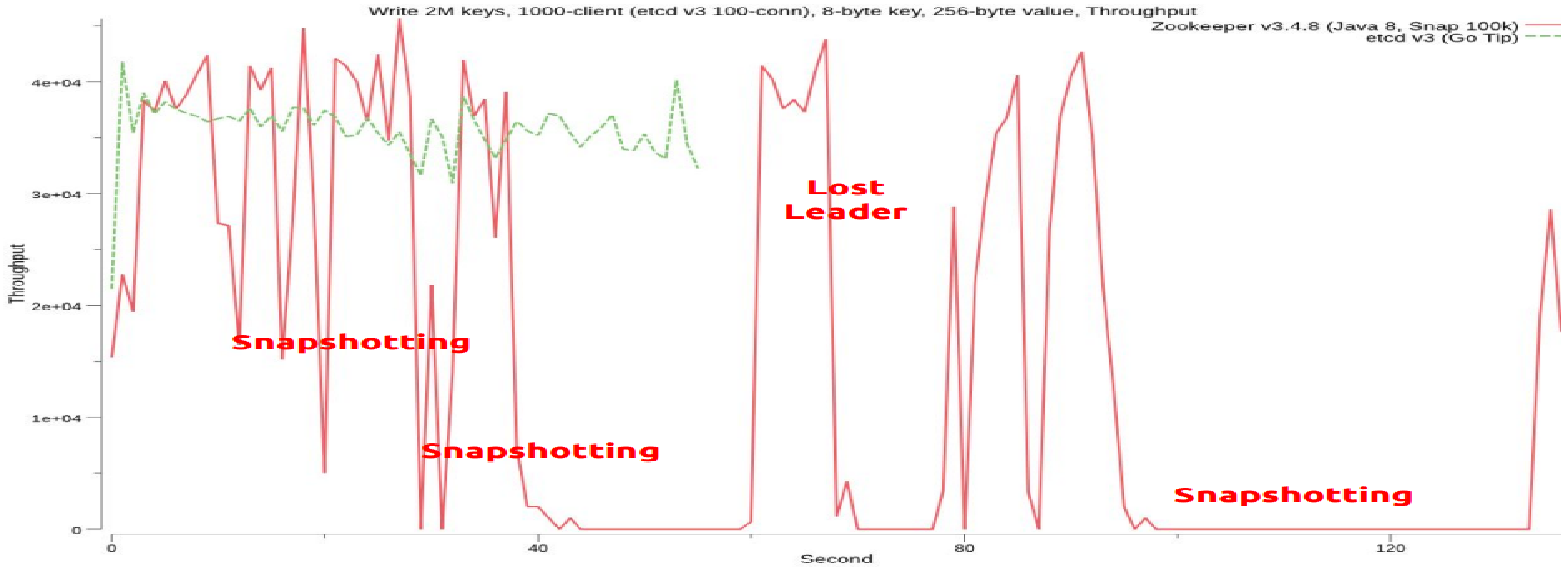

Write 2M keys, 1000-client (etcd v3 100-conn), 8-byte key, 256-byte value, Memory(MB)
Zookeeper v3.4.8 (Java 8, Snap 100k)
etcd v3 (Go Tip)
etcd v2 (Go 1.4)
Memory(MB)
3000
2000
1000
0
100
200
300
400
500
Second

etcd Project Status: Reliability

Throughput


Write 2M keys, 1000-client (etcd v3 100-conn), 8-byte key, 256-byte value, Throughput
Zookeeper v3.4.8 (Java 8, Snap 100k)
etcd v3 (Go Tip)
Throughput
0
1e+04
2e+04
3e+04
4e+04
0
40
80
120
Second

etcd Reliability



ZooKeeper logs

```
07:16:35 [Snapshot Thread:FileTxnSnapLog@240] - Snapshotting...
07:16:43 fsync-ing the write ahead log in SyncThread:3 took 1224ms...
07:16:46 fsync-ing the write ahead log in SyncThread:3 took 3205ms... // Snapshotting
...
07:17:14 [FastLeaderElection@818] - New election...                  // Leader Election
```

etcd Reliability

Functional tests dash.etcd.io (http://dash.etcd.io/dashboard/db/functional-tests)

- Kill one/all members
- Kill leader
- Network partition
- Network latency

Consistency checking after recovery

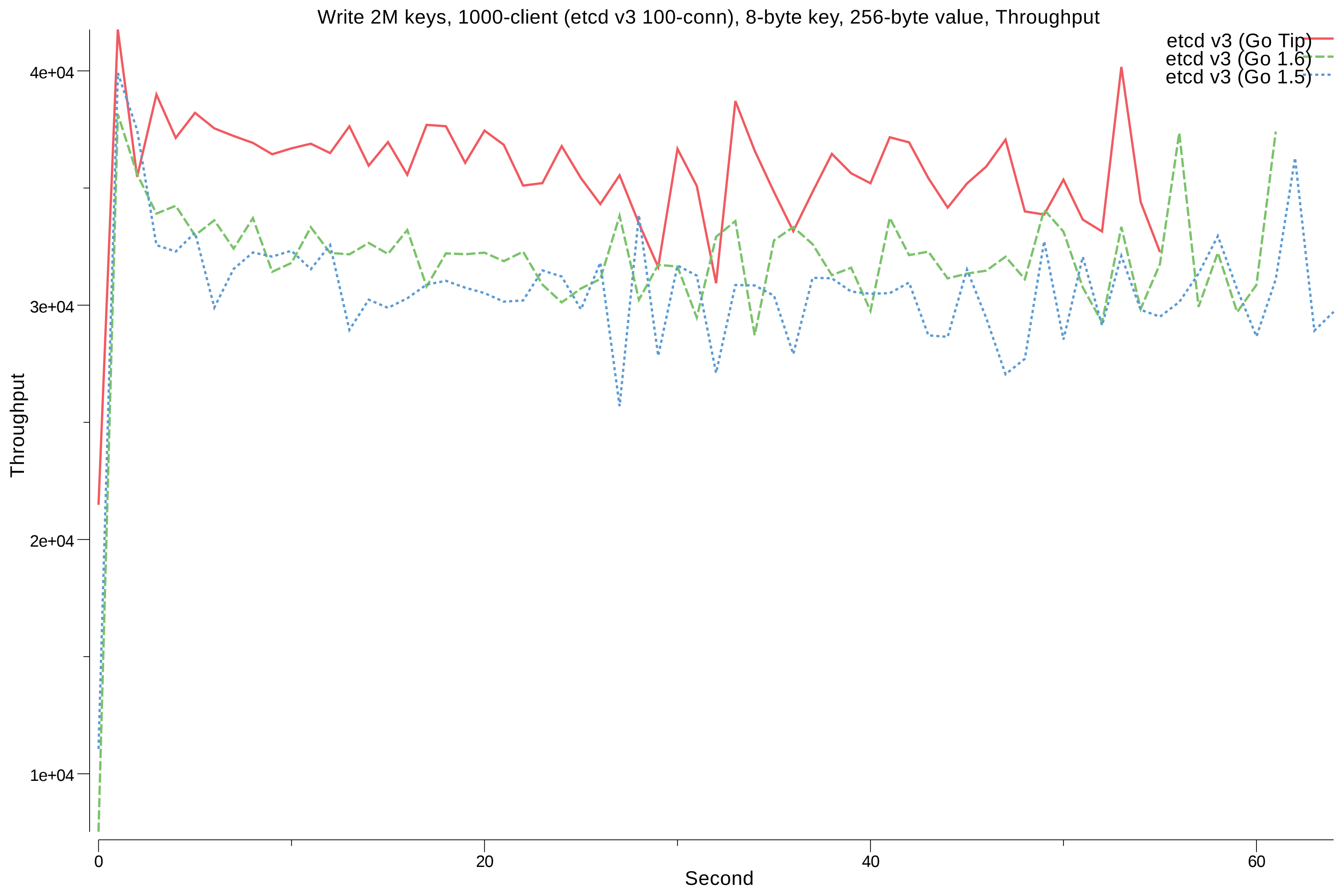
>12,000 failure injections per day

>2M injected for etcd v3

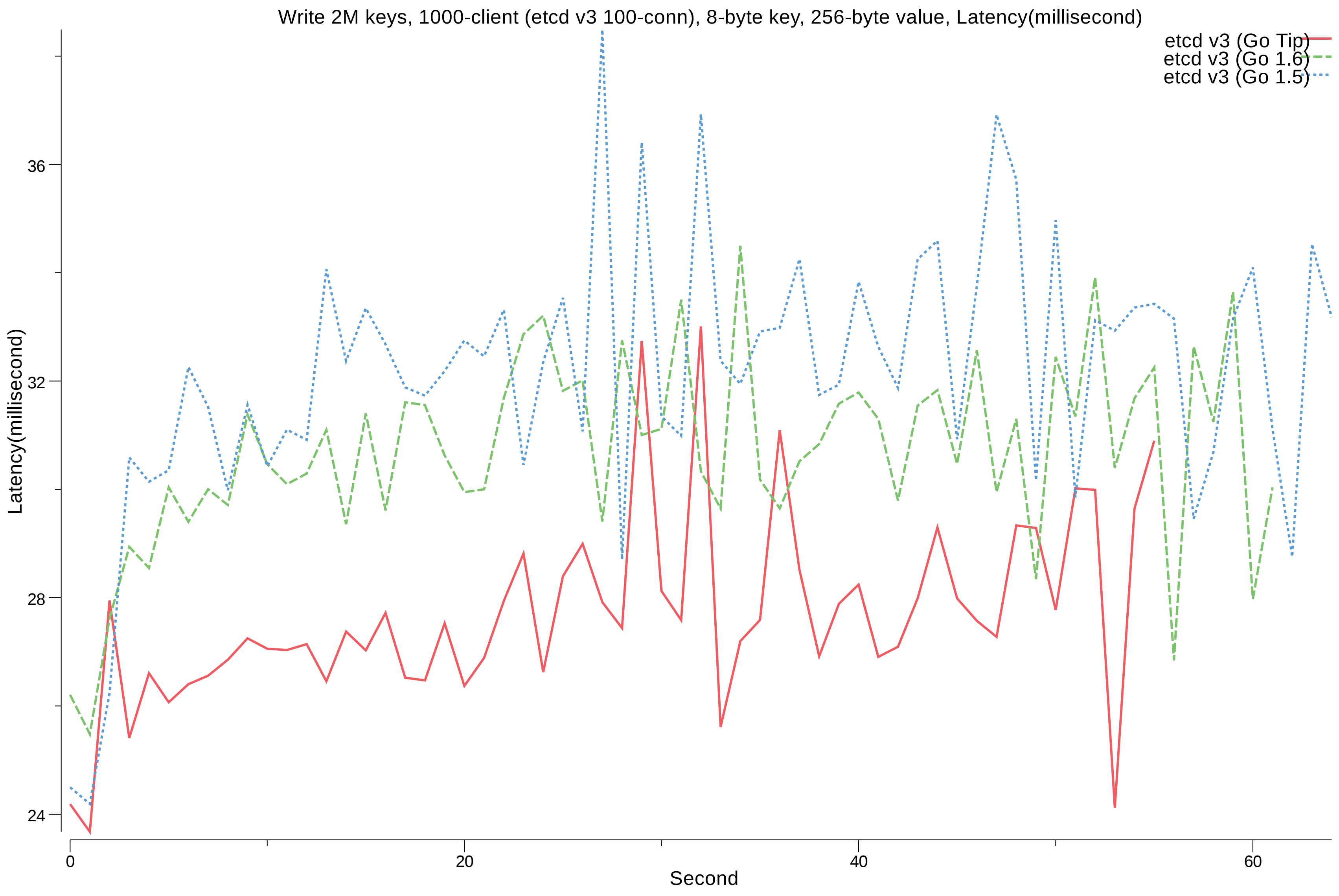
Go 10 Tips

#1 Use latest Go

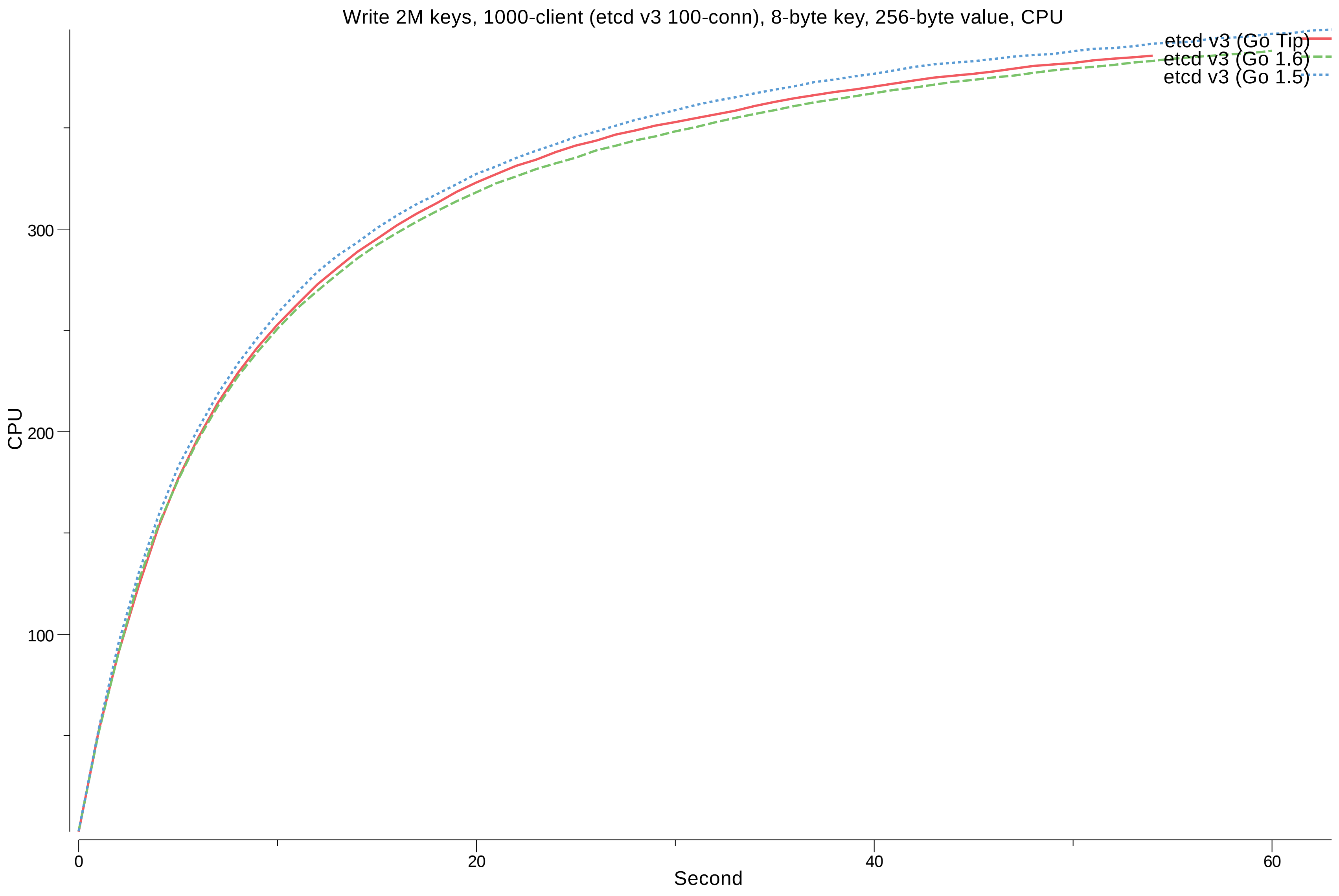
Throughput


Write 2M keys, 1000-client (etcd v3 100-conn), 8-byte key, 256-byte value, Throughput
etcd v3 (Go Tip)
etcd v3 (Go 1.6)
etcd v3 (Go 1.5)
Throughput
4e+04
3e+04
2e+04
1e+04
Second
0
20
40
60

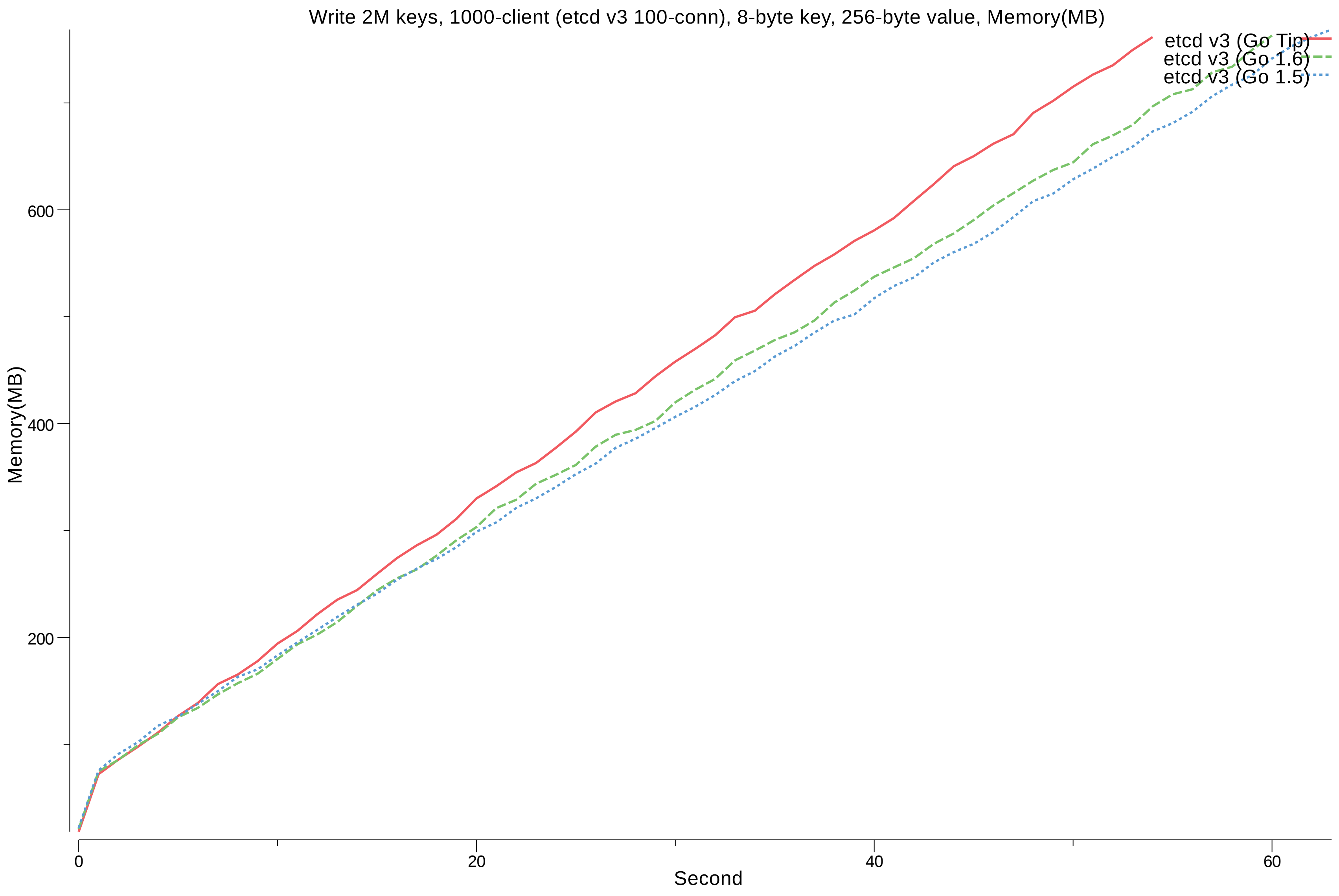
Latency


Write 2M keys, 1000-client (etcd v3 100-conn), 8-byte key, 256-byte value, Latency(millisecond)
etcd v3 (Go Tip)
etcd v3 (Go 1.6)
etcd v3 (Go 1.5)
Latency(millisecond)
36
32
28
24
0
20
40
60
Second

CPU


Write 2M keys, 1000-client (etcd v3 100-conn), 8-byte key, 256-byte value, CPU
etcd v3 (Go Tip)
etcd v3 (Go 1.6)
etcd v3 (Go 1.5)
CPU
300
200
100
0
20
40
60
Second

Memory


Write 2M keys, 1000-client (etcd v3 100-conn), 8-byte key, 256-byte value, Memory(MB)
etcd v3 (Go Tip)
etcd v3 (Go 1.6)
etcd v3 (Go 1.5)
Memory(MB)
600
400
200
0
20
40
60
Second

#2 Check slice allocation

Go slice capacity decides how much memory to use in the backing array

More capacity means more allocation.

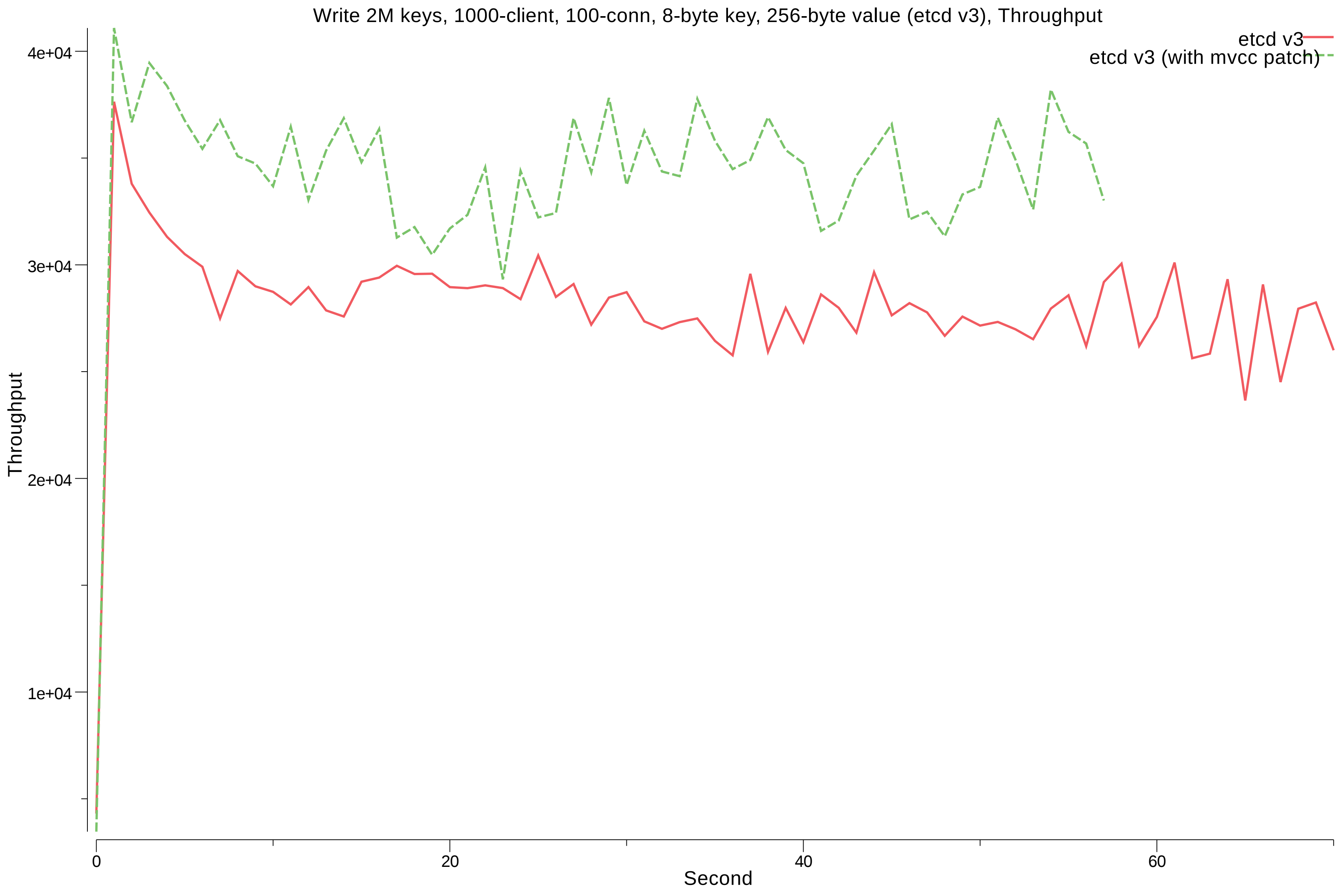
Don't allocate more than you need.

GH5238 (https://github.com/coreos/etcd/pull/5238)

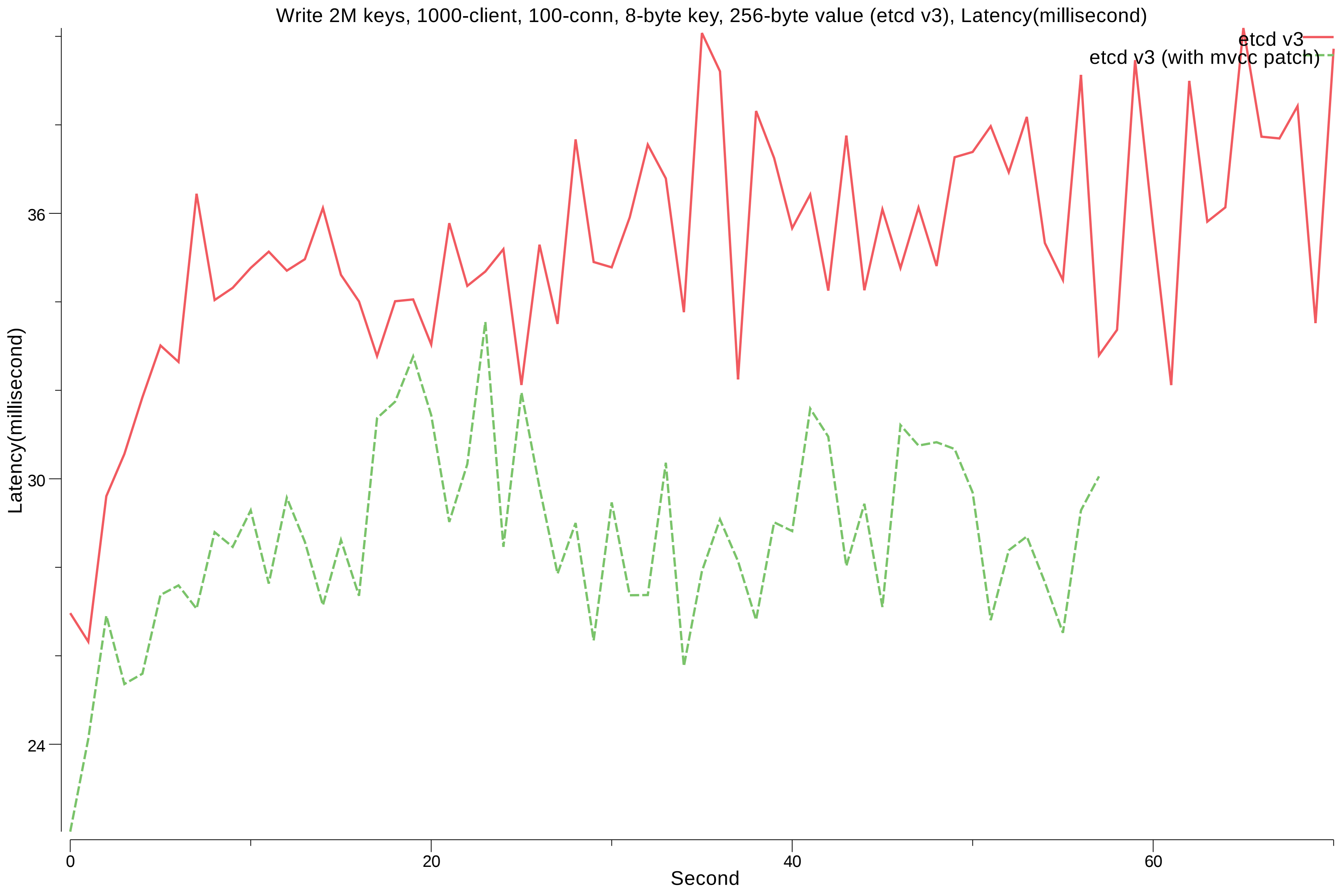
```
s.changes = make([]mvccpb.KeyValue, 0, 128)
s.changes = make([]mvccpb.KeyValue, 0, 4)    // better for etcd use case
```

Check your use case when making slice!

Throughput


Write 2M keys, 1000-client, 100-conn, 8-byte key, 256-byte value (etcd v3), Throughput
etcd v3
etcd v3 (with mvcc patch)
Throughput
4e+04
3e+04
2e+04
1e+04
0
20
40
60
Second

Latency


Write 2M keys, 1000-client, 100-conn, 8-byte key, 256-byte value (etcd v3), Latency(millisecond)
etcd v3
etcd v3 (with mvcc patch)
Latency(millisecond)
36
30
24
0
20
40
60
Second

#3 Test your code

All with "go test" command

- unit tests
- integration tests
- functional tests
- benchmarks

Use expect package for e2e tests

```
proc, _ = expect.NewExpect("etcdctl", "get", "foo")
_, err = proc.Expect("bar")  // if err != nil, found a bug!
```

github.com/coreos/etcd/pkg/expect(https://godoc.org/github.com/coreos/etcd/pkg/expect)

#4 Check goroutine leaks

Scan runtime.Stack

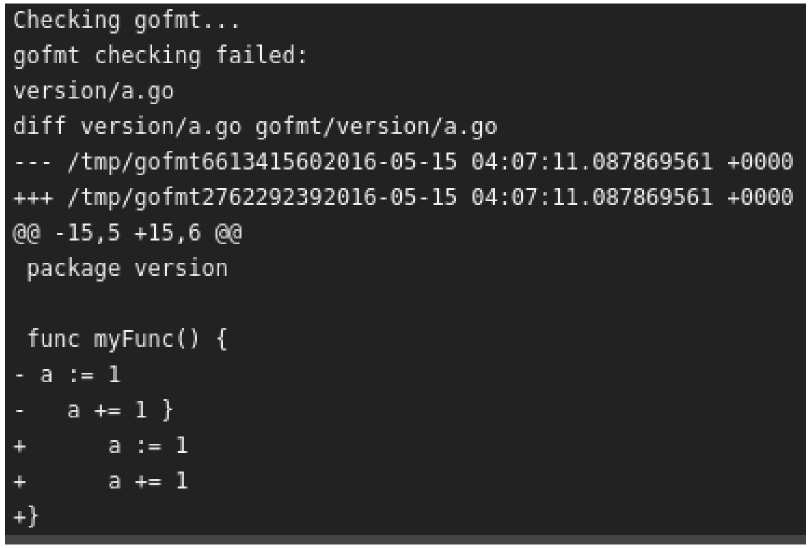
```
func TestMain(m *testing.M) {
    v := m.Run()
    if v == 0 && testutil.CheckLeakedGoroutine() {
        os.Exit(1)
    }
    os.Exit(v)
}

func TestSample(t *testing.T) {
    defer testutil.AfterTest(t)
    ...
}
```

- net/http/main_test.go (https://github.com/golang/go/blob/master/src/net/http/main_test.go)
- github.com/coreos/etcd/pkg/testutil (https://godoc.org/github.com/coreos/etcd/pkg/testutil)

Highly recommend for projects with context.Context, gRPC

#5 Always gofmt, go vet

gofmt



```
Checking gofmt...
gofmt checking failed:
version/a.go
diff version/a.go gofmt/version/a.go
--- /tmp/gofmt6613415602016-05-15 04:07:11.087869561 +0000
+++ /tmp/gofmt2762292392016-05-15 04:07:11.087869561 +0000
@@ -15,5 +15,6 @@
 package version

 func myFunc() {
- a := 1
-   a += 1 }
+      a := 1
+      a += 1
+}
```

go vet

```
log.Fatalf("hello %d", "a")
// arg "a" for printf verb %d of wrong type: string
```

#6 Write simple Go

- github.com/dominikh/go-simple (https://github.com/dominikh/go-simple) by Dominik
- Simplify your Go code

```
ok := true
if ok == true {}  // X
if ok {}          // O
```

Don't:

```
err := l.newStream()
if err != nil {
    return err
}
return nil
```

Do:

```
return l.newStream()
```

#7 Check unused

- github.com/dominikh/go-unused (https://github.com/dominikh/go-unused) by Dominik
- Finds unused constants, variables, functions and types

```
func reportMetrics() {}
// func reportMetrics is unused
```

Found bugs in etcd GH4955 (https://github.com/coreos/etcd/pull/4995/files)

#8 Use goword

- github.com/chzchzchz/goword (https://github.com/chzchzchz/goword) by Anthony (etcd team)

Comment checker

```
// This.
func Hello() {}  // This. (godoc-export: This -> Hello?)"
```

Spell checker

```
// Hello retuens.
func Hello() {}  // Hello retuens. (spell: retuens -> returns?)
```

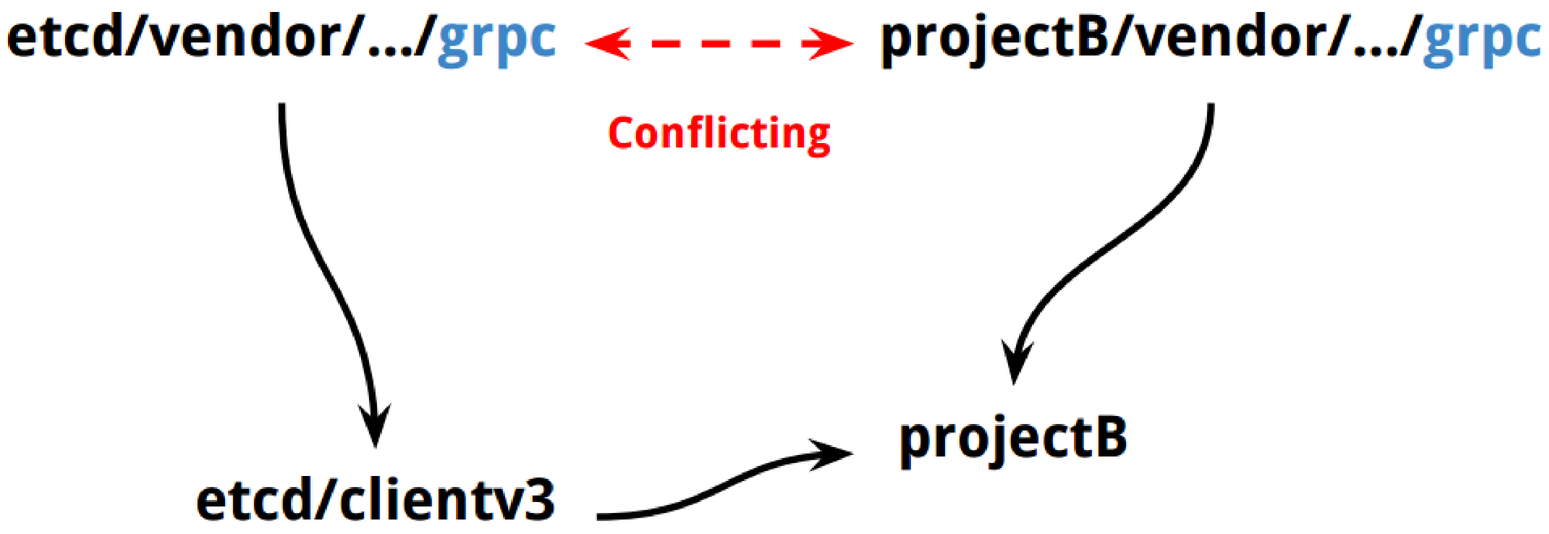
goword understands your Go syntax

#9 Document with godoc

- etcd must be easy to use
- etcd needs good documentation
- Use godoc to document our client package with code examples
- etcd/clientv3 (https://github.com/coreos/etcd/tree/master/clientv3)

#10 vendor

etcd use case is complicated...



#10 vendor

Problem

- etcd "client" package is used within etcd repo (etcdctl)
- etcd "client" imports "grpc" and vendors it
- Project B imports this etcd "client" package
- Project B also uses "grpc", but from different import path

Now two projects has conflicting "grpc" code GH566 (https://github.com/grpc/grpc-go/issues/566)

```
panic: http: multiple registrations for /debug/requests
```

#10 vendor

Solution GH4950 (https://github.com/coreos/etcd/pull/4950)

- Create symlinks inside cmd directory
- ln -s main.go cmd/main.go
- cmd/vendor

To update dependency

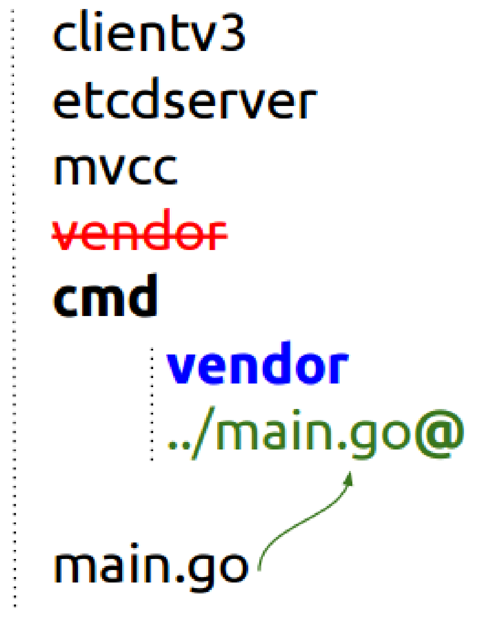
```
ln -s cmd/vendor vendor && godep save
```

#10 vendor

Still go-get-able.

No conflicts with other projects.

Works, even on Windows!



Thank you

Gyu-Ho Lee
CoreOS
gyu_ho.lee@coreos.com (mailto:gyu_ho.lee@coreos.com)
https://github.com/coreos/etcd (https://github.com/coreos/etcd)