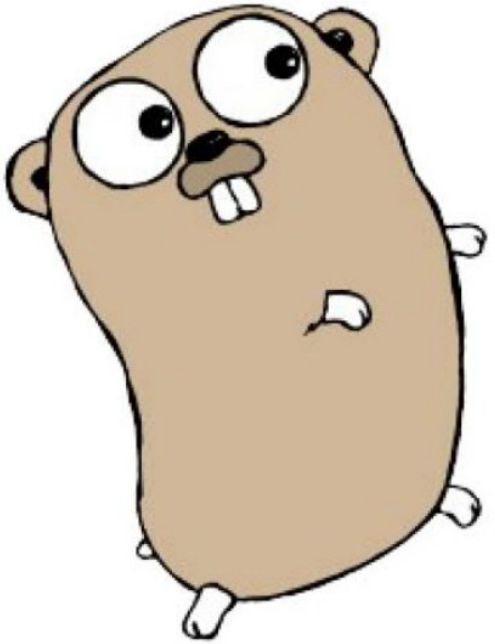
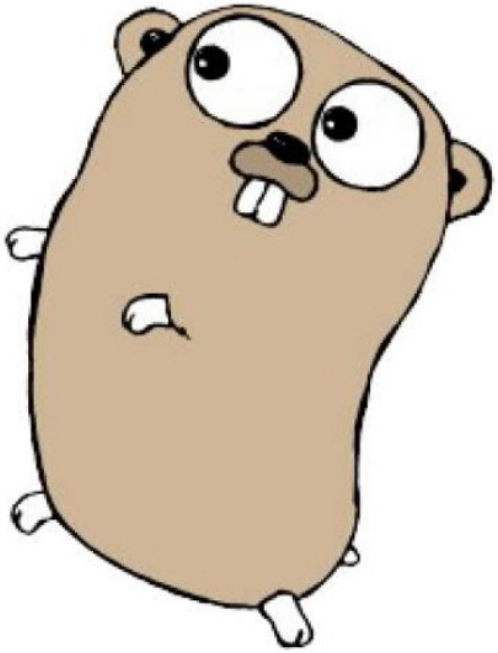
Scalability improvements and benchmark for your NoSQL database – tiedot



Howard Guo



What is tiedot?

- Finnish word - “data”
- Document database engine implemented in Go
- Flexible API – choice of HTTP or embedded
- High throughput – thrashing competitions by more than 200% in benchmarks
- https://github.com/HouzuoGuo/tiedot
  (or just google the name)

  And now...
- High scalability

The scalability challenge

- The Go runtime has trouble scaling beyond 8 processor cores
- Also it does not scale across CPU sockets

… for computation-heavy workloads of DB server

The scalability goal

Scale beyond a single CPU socket!

- Split the single database process into multiple server processes
- Use Unix domain socket for IPC

Hackweek Achievement

- The IPC implementation was at 80% completion last month
- … and 100% completed during the Hackweek.
- Tweak and implement the throughput benchmark.

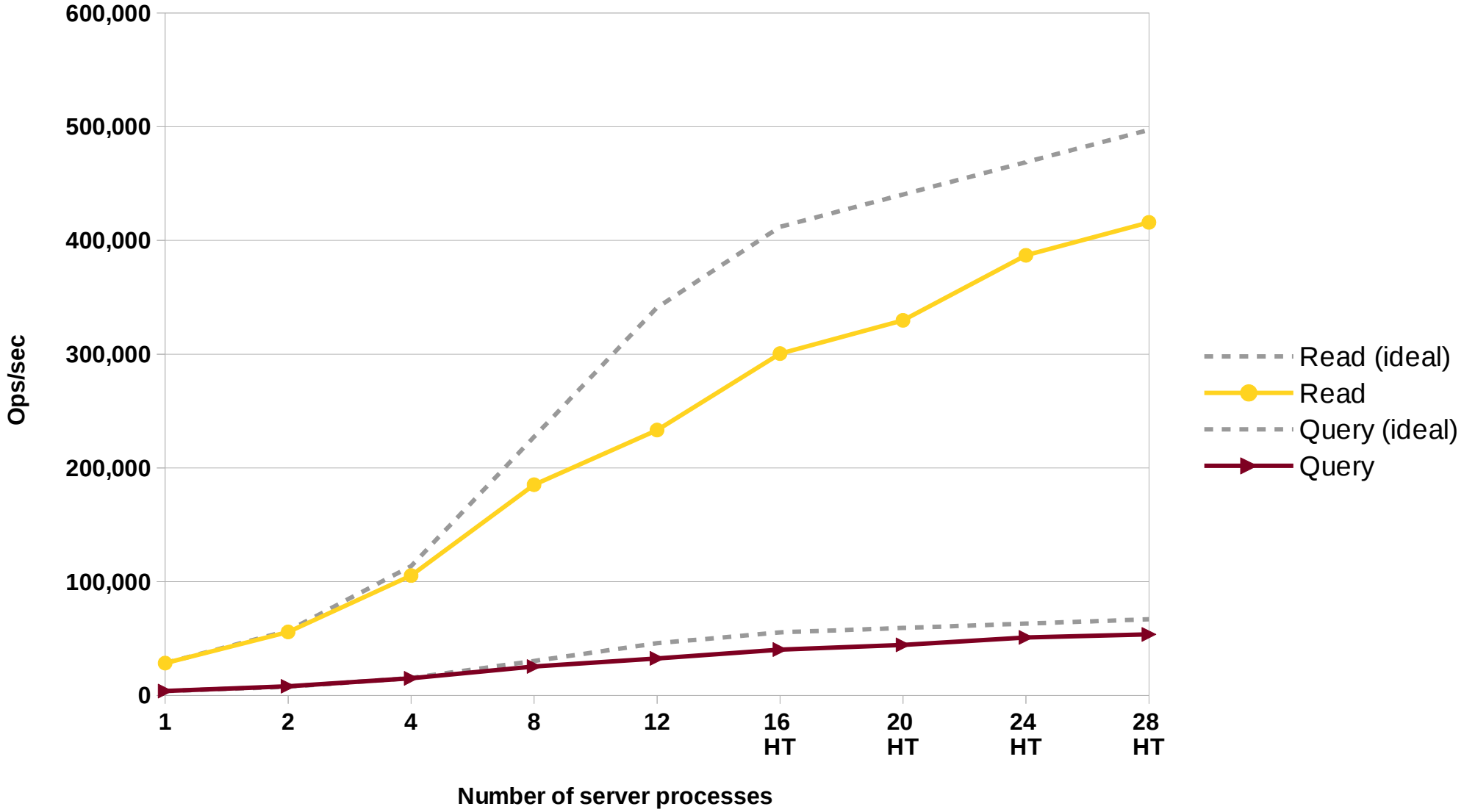
Now let's get into the benchmark!

Benchmark Environment

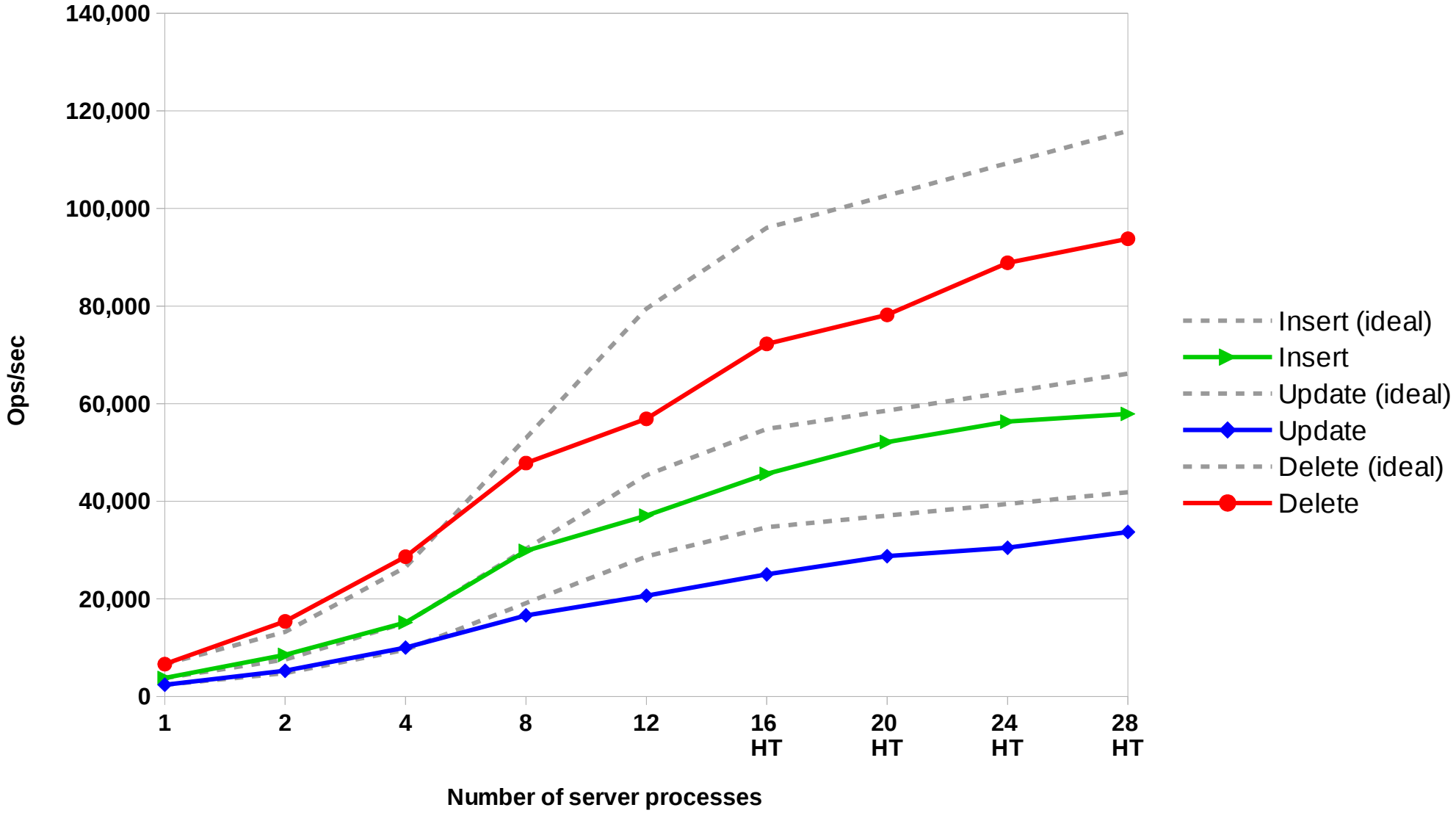
- S(calable)USE Linux Enterprise 12
- Intel Xeon E5-2697 v3 - dual socket
- 32 GB DDR4

- `./tiedot -mode=ipc-bench -gomaxprocs=N` launches N server processes and the same number of benchmark clients.

Throughput – Read Ops



Throughput – Write Ops



Lessons learnt

- Unix domain socket is efficient and scalable
- SUSE Linux is scalable (we knew)
- CPU affinity never ceases to boost performance of computation workloads

Extensive usage of mmap favours:

- Pre-allocated data files, no holes and nodatacow.
- Low `vm.swappiness`
- Reasonably high `vm.dirty_ratio/bytes`
- Reasonable `vm.dirty_background_ratio/bytes`
- When the total size of mmap exceeds a certain value, btrfs will disobey `vm.dirty_ratio/bytes`
  A feature, perhaps?

Thank you

Yuko Honda ©

Questions?