Examining single server database systems in the context of real-time sensor data
WeBike Project
WeBike Project

• Current status:
  • **28 e-bikes** among faculty/students/staff
  • Equipped with **sensors**:
    • GPS
    • Accelerometer
    • Temperature (different locations)
    • (Dis)charging current
    ...

• Data collection:
  • **1/min** for detecting charging/discharging event
  • **1/sec** during charging/discharging event
WeBike Project

• Data transfer:
  • Upload log files to server via WiFi
  • Insert data into MySQL database via cron job
WeBike1000 Project (Future)

- Distribute at least **1000** e-bikes
- Transfer data **instantaneously**
- Allow for **real-time** analysis
WeBike1000 Project (Future)

- Estimating size of data:
  - $1,000 \times 84,600 = 84,600,000$ rows/day
  - Data size/row: $\sim 0.5$ kB
  - $> 40$ GB/day

$\Rightarrow$ **MySQL** probably **not adequate** anymore
Project idea

• **Huge** clusters/cloud/data centers **not needed**
  • Academic environment
  • “Not everybody is Google or Facebook”

• Examine **performance** of **small scale** database systems
Project objective

- Create a **benchmark**
  - Representative of load & queries

- **Compare** different **database** systems on **target hardware**
  - Single server
  - Big but limited RAM (<< 1 TB)
  - SSD as “long” term storage (data will be archived on regular HDD)

- Software **freely available** (at least in academic context)
Current status

• Actual server occupied until next week
  → Set up Ubuntu VM for testing

• Installed kdb+ column-store
• Created webike database
• Running data collection simulation & database insert
• Column store

• Databases can be partitioned (e.g. by date)

  webike/
  2016.07.18/ sensorData/ bikeNo.q
  2016.07.19/ time.q
  2016.07.20/ gpsLat.q

• Multiple indices on different columns possible
  • Add attributes to bikeNo, time
kdb+

- Works both in **main memory and on disk**
  - Accumulate data & flush to disk at end of day
  - Queries work on both

- Uses **q** query language
  - Includes commands similar to SQL
  - Interfaces to most major programming languages (Java, C#, C++, Python…)
  - MapReduce capabilities

- Fast and low footprint
Future work

• Set up a **benchmarking** framework

• Compare to other database systems (**Spark** stack, **MySQL**…)

• Run the tests on the **server** instead of a local VM

• **Generalize** findings for less specific circumstances
Thank you!