Data Persistence Project Group (Inaugural) Meeting – 8/14/18

Attendees: Brett, Michael (LF), Alberto (NOV), Emad (), Fede (Cavium), Itamar Haber (Redis), Janko (Mainflux) Trevor, Chandra, Jim, Tom Pool, Jeroen Mackenbach, Emad Attia, Eric (Dell), Bruce Huang , Edward Scott, Andy, Steve (IoTech), Markus (ObjectBox). Attendees that may have joined after the start of the meeting may not have been captured and listed.

Discussion and action items as a result of meeting in RED

Old Business

- New meeting time was based on consensus vote
    - No objections – meeting time will stand for future meetings
- Continuation of requirements discussion
    - NOV requirements (oil&gas)
        - 5 to 50K writes per second (depends on use case / device)
        - Majority are between 100-3000 on average writes per second
            - Size: Kilobytes per message
        - Reads are 10-20% of writes; while not a large number of reads, these do need to be efficient
            - Interest in workloads – what you are reading may be more important than just a simple read (get all reads or events)
        - Size of database on disk is not really an issue
        - RAM and CPU usage are the biggest concerns (Mongo gave them problems on these)
        - Startup time was also a concern (again Mongo problem)
        - Data is stored for up to 6 months on the edge (they have means to deal with too much or lost data at the edge)
        - Prefer to not embed the database – prefer flexibility to change and to distribute elsewhere
        - They don't have binary data, but use protobuf for Readings.
    - Hitachi Ventara
        - high ingress data @0.1 sec/message
    - Per last meeting
        - RFID Use case: 100+ / sec sensor reads
        - Building automation use cases: 1000+/sec
    - From Chandra Venkatapathy @ Dell

Data source	Sampling Rate	Order of Volume of data collected in s	Column1	
Line Fault detection in electric grid	in Micro second	Bytes, Kilo Bytes	Usually an interface betwe	
	Response in second			
Vibration Sensors	in msec/ sec	Bytes, KB		
Pressure Sensors	in msec/ sec	Bytes, KB		
PLC	in msec/ sec	Bytes, KB		
CNC Machines	in msec/ sec	KB, MB		
CAN Bus	in msec/ sec	MB		
Building Automation Applications	in msec/ sec	Bytes, KB		
Locomotive Telemetry	in msec/ sec	KB, MB		
RTU in Utility	in msec/ sec	KB, MB		

    - Current Must haves

- License of product – compliant with Apache 2
- Store and forward needs
- Platform support:
  - Intel , ARM 64 bit
  - OS support: Linux, Unix, Windows, MacOS – those EdgeX has targeted
- Support for batch writes is important
  - But this might require API changes
- Performance
  - Need a holistic view. Small but slow is not acceptable.
  - Memory size, footprint, CPU, network? What is size? All these must be taken into consideration together.
  - Typically Prioritize writes over reads in performance
  - Concern impact of backup processes
- Durable across EdgeX shutdown
- Run in a container (Docker/Kubernetes/Snap/etc.)
  - Has to manageable from one control plane
  - Embedded might be considered by some, flexibility is more important to most (ability to easily swap out the database with proper app abstraction), and ability to distribute to alternate platform from micro services
- Secure
  - Password protected
  - Supports data encryption (protect data at rest)
  - At least what we have in Mongo
- Has Java, Go, C, C++ drivers/connectors
- Community support and user-base size
  - How to quantify?
  - Number of github stars
  - Enterprise deployment support/support for commercial deployment (nice to have)
- Binary support (as long as we can identify what type of binary)
  - Max size (up 16MB)
- Most use cases store data for day, a week not months
  - Capture quickly and ship it off
  - Provide enough data for historical back look for local actuation
  - Up to 10M data points on average; 100M max
  - Use case would dictate platform and architecture to deal with more or less

- Nice to haves
  - User administration / usability
    - Has ability to integrate to identity management
  - NoSQL (versus SQL) – assumed NoSQL given data types/objects; but immaterial otherwise
    - From Eric Cotter post meeting: Wouldn't you want to at least bench mark SQL in some fashion to show it as a contrast "not justification per say" for using a NoSQL DB? A lot of the commercial players may want to see that for justifying purchasing a NoSQL.
  - Synch capability

            - Careful not to exclude non-enterprise solutions with this nice to have
        - Backup support
            - Again, careful not to limit to enterprise systems
        - Transactional (ACID) or Eventual Consistent (which CAP axis)?
            - Availability and partition tolerance are priority
            - Totally use case dependent
        - Support multi-tenancy
            - Again use case/market driven
            - May not apply at the edge
        - 32 bit support (Intel or ARM)
        - Database that also runs in memory
- Should we start collecting potential options and can someone lead that effort
    - Could we have some sort of use case dependent data feed to simulate how the database perform; test harness
        - Could we get implementers of PRs to volunteer to help setup that test harness per database of their choice
        - Topic for next week – how to evaluate
    - Not a review or evaluation at this stage, just a list of potential databases
        - MongoDB (use as our baseline)
        - ~~Mongo Mobile~~
        - Redis
        - ~~Influx~~ – may not be general purpose; wouldn't support something like metadata, export, etc.
        - ~~Cassandra~~ –heavy? Needs 1GB of memory; more of a cloud target
        - CouchDB
        - Couchbase – research on which version
        - ObjectBox
        - If we have volunteer to do work then look at SQL options
            - Postgres
            - MySQL
            - SQLite
            - CockroachDB – is this a cloud SQL database; can't run in single node config?

New Business

- We need benchmarks/performance testing – at data level; on the simple and complex queries, multiple types of writes, etc.
- Need test on multi-threaded , concurrent uses – simulating multiple clients (load tests)