Incorporating Blockchain into RDF Store at The Lightweight Edge Devices

Anh Le-Tuan, Darshan Hingu, Manfred Hauswirth and Danh Le Phuoc | Technische Universität Berlin

RDF4LED - An RDF Engine for Edge Devices

Incorporating Blockchain into RDF4Led

Why Edge Devices ???

Edge Computing - Internet of Things

- reduce network overhead.
- reduce latency for real-time applications.
- improve scalability.
- better privacy control.

Linked Data

- enable data integration of heterogeneous sources.
- enable data federation over edge nodes.

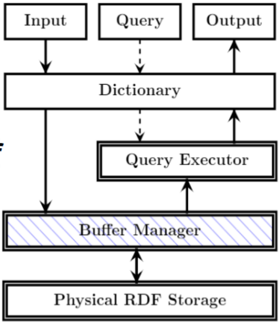
Missions

Moving semantic data processing task away from centralised cloud for the IoT

- How much semantic data on small devices ?
- How to scale data federation over small devices on edge systems ?
- How to encourage people sharing the data from their edge devices ?

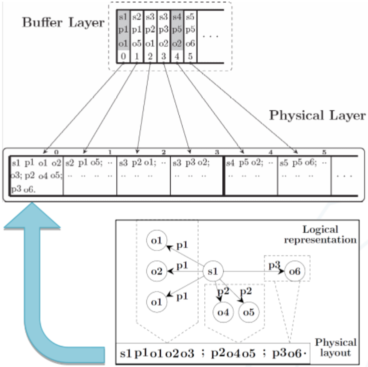
System Design

- **<u>Physical storage</u>:** Introduce a ***two-layers index schema*** for triples to adapt the ***flash I/O behaviors***.
- **<u>Buffer Manager</u>:** Introduce a ***buffer replacement policy*** to reduce ***the number of overwritten***.
- **<u>Query Executor</u>**: Introduce an ***adaptive strategy*** for ***iterative join execution*** to avoid caching ***intermediated results*** that ***minimizing memory usage***.



Technische Universität Berlin

Physical RDF Organisation: Two-Layers Index



Buffer Layer:

- managing triples **in memory**.
- acting as ***an index*** for the Physical Layer by keeping ***the first triple*** of a page and mapping it to ***the physical address*** of the page.
- organising the ***atomic triples*** into pages.
- grouping the ***dirty pages*** into writing blocks.

Physical Layer:

- Managing triples **on flash**.
- Using **molecule-based data structure** to avoid storing **repetitive** values.
- Keeping the molecules continuously sorted in pages, and keeping the pages sorted in blocks.

Open Distributed Sytems ODS

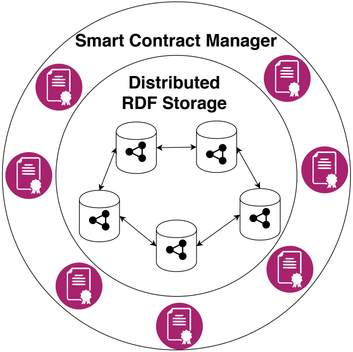
RDF4LED (cont.)

Targeted Small Devices

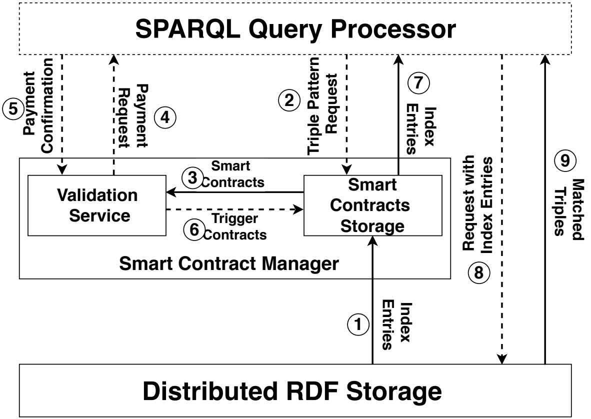
	Devices		
	Pi0	BBB	GII
CPU	ARM 11, 1.0 GHz, 1-core	ARM A8, 1.0 GHz, 1-core	x86 Quark, 0.4 GHz, 1-core
RAM	512 MB	512 MB	256 MB
Storage	Transcend MicroSD 16GB class 10 (40MB/s)		
OS	Raspbian	Debian 7.0	Yocto

Incorporating Blockchain into RDF4Led

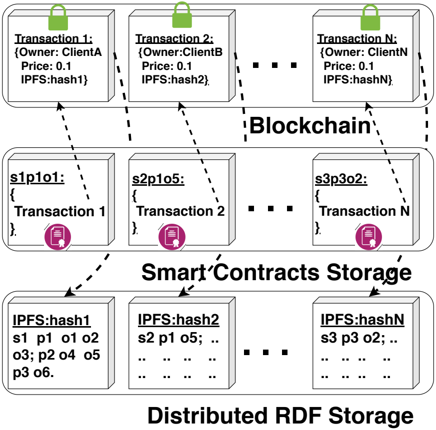
System Overview



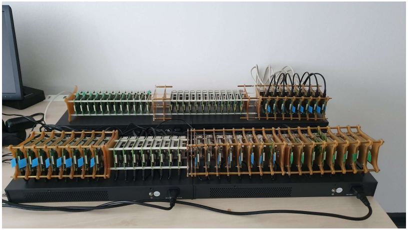
System Workflow

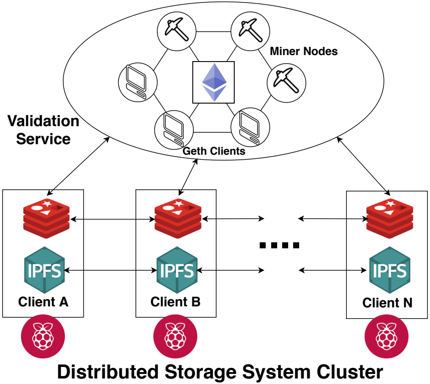


Physical Organisation

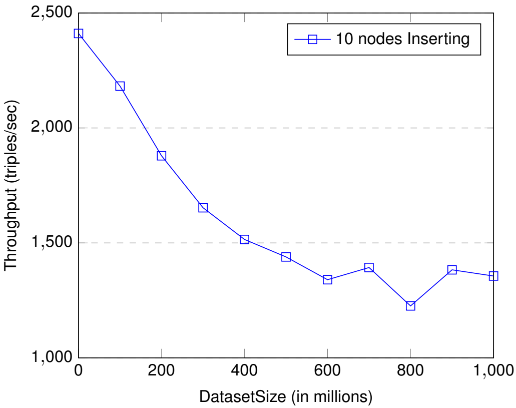


System Deployment

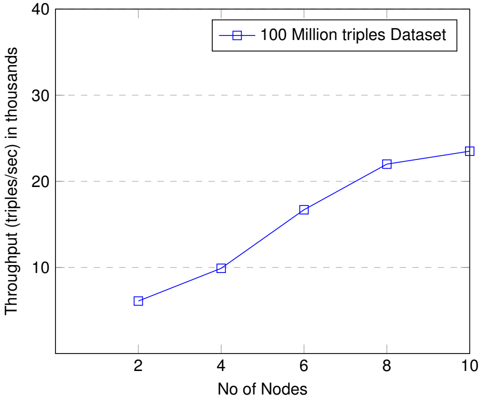




Evaluation - Input (1) Acc. Throughput on Static Cluster Sizes

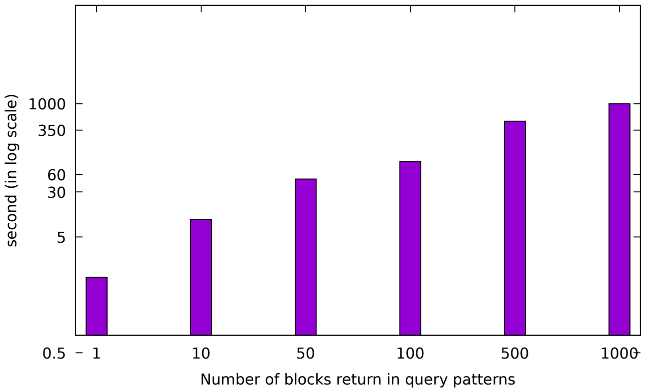


Evaluation - Input (2) Acc. Throughput on Varying Cluster Sizes



Technische Universität Berlin

Evaluation - Query

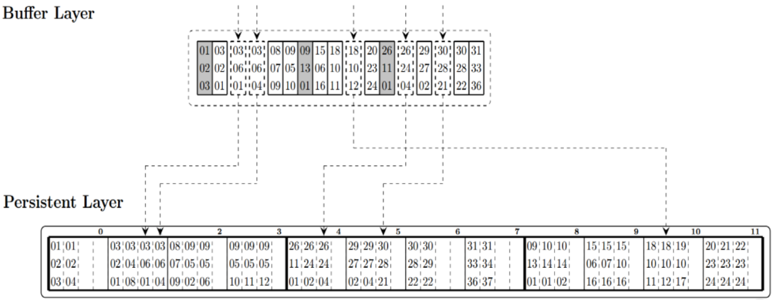


Open Distributed Sytems ODS

Thank You !!!

RDF4LED (cont.)

Write Management (1): Clustering example



Write Management (2): Cache Management

Replacement policy:

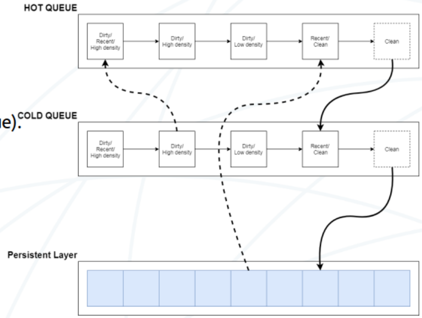
- data access frequency.
- probability of data being changed in future.

Cache organisation:

- using 2 buffer queues (hot queue/cold queue).
- the recent active blocks are kept in the hot queue.
- the inactive blocks are kept in the cold part.

Blocks are sorted in queue with following priority:

(i) the clean/unmodified blocks;
(ii) the number of atomic triples in a block
(iii) the higher density blocks :
(iv) the last recently access block.



Replacement Schema

RDF4LED (cont.)

Push-based Join Algorithm

Algorithm 1: Join propagation

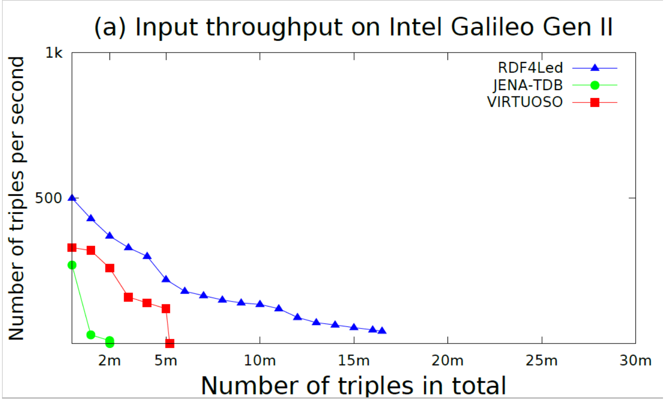
Function `propagate` $(\mu, \mathbb{P})$
input : μ : mapping, $\mathbb{P}$: set of triple query patterns
output: μ : mapping
if *isEmpty*($\mathbb{P}$) **then**
  return μ;
$p \leftarrow$ `findNextPattern` $(\mu, \mathbb{P})$;
$p_{key} \leftarrow createKey(\mu, p)$;
$\mathbb{T} \leftarrow$ indexScan(p_{key});
$\mathbb{P}' \leftarrow \mathbb{P} \setminus \{p\}$;
for $t \in \mathbb{T}$ **do**
  $\mu \leftarrow$ `bindMapping` (t, p);
  `propagate` $(\mu, \mathbb{P}')$;
  $\mu \leftarrow$ `resetMapping` (t, p);

Algorithm 2: Find the next triple pattern

Function `findNextPattern` $(\mu, \mathbb{P})$
input : μ : mapping, $\mathbb{P}$: set of triple query patterns
output: P : triple query pattern
$p_{next} \leftarrow null$;
$s_{min} \leftarrow Integer_{max}$;
for $p \in \mathbb{P}$ **do**
  if *isShared*(μ, p) **then**
    $p_{key} \leftarrow createKey(\mu, p)$;
    $\mathbb{I} \leftarrow indexLookUp(p_{key})$;
    $s \leftarrow sizeOf(\mathbb{I})$;
    if $s < s_{min}$ **then**
      $s_{min} \leftarrow s$;
      $p_{next} \leftarrow p$;
return p_{next};

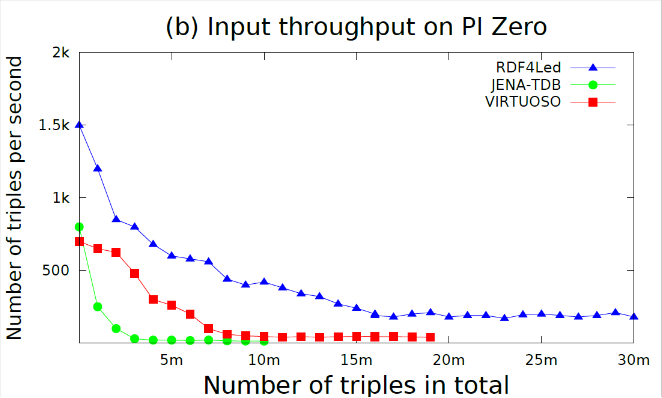
RDF4LED (cont.)

Evaluation Results - Input (1)



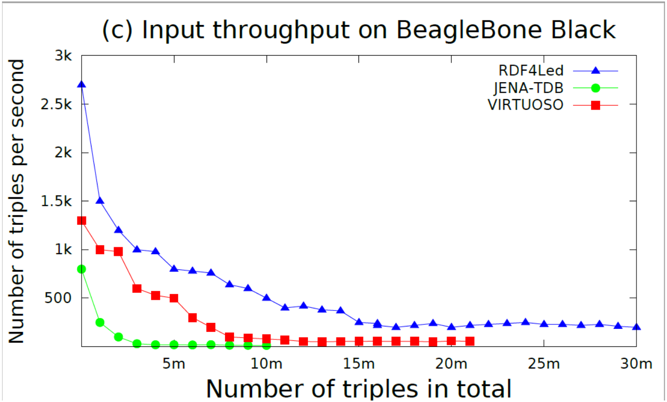
RDF4LED (cont.)

Evaluation Results - Input (2)



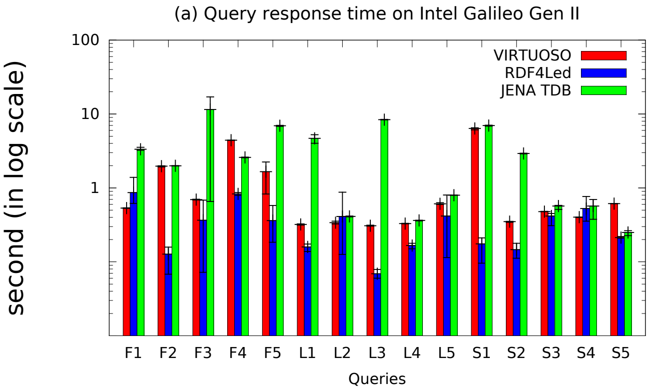
RDF4LED (cont.)

Evaluation Results - Input (3)



RDF4LED (cont.)

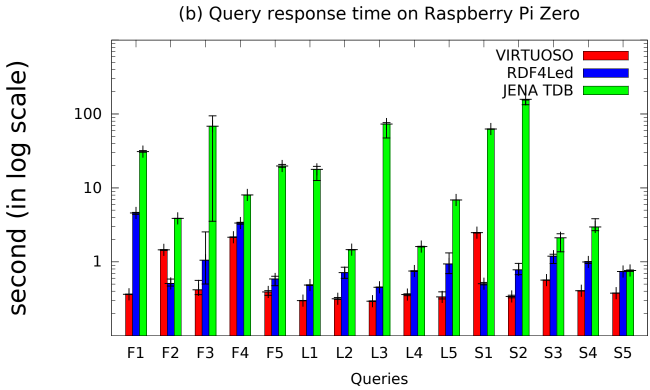
Evaluation Results - Query (1)



RDF4LED (cont.)

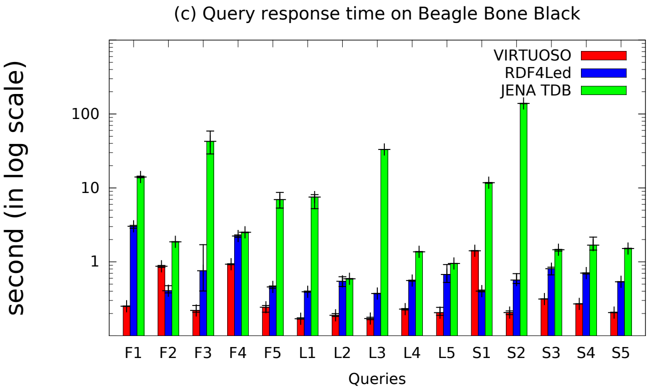
Technische Universität Berlin

Evaluation Results - Query (2)



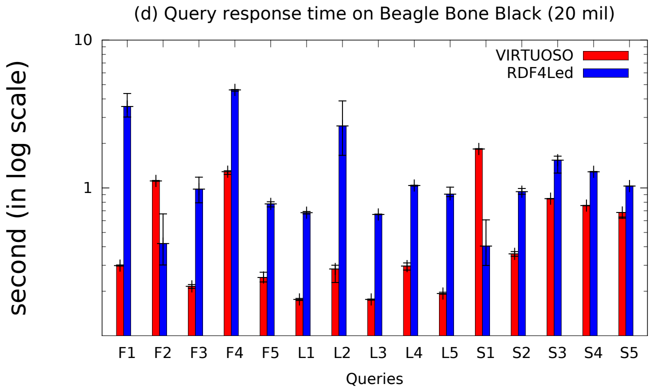
RDF4LED (cont.)

Evaluation Results - Query (3)



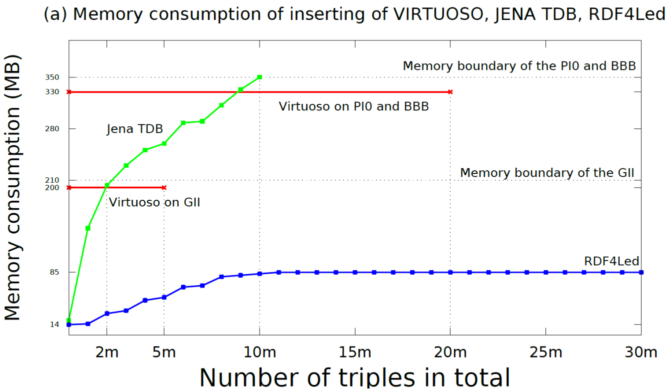
RDF4LED (cont.)

Evaluation Results - Query (4)



RDF4LED (cont.)

Evaluation Results - Memory (1)



RDF4LED (cont.)

Evaluation Results - Memory (2)