

Dokumentation T2 „Fluxen“

Fachbereich Automatisierung/Informatik

Hochschule Harz, Wernigerode

VT I 4.0

Prof. Adler

Norman Bauersfeld

Studiengang bbgl. DaSc(M. Sc.)

Matrikel 30315

Vorraussetzungen

- InfluxDB; Installation einer Datenbank für das Speichern von Zeitreihen
- Datensatz mit Stromdaten der Messung „shellies“
- Datensatz mit Anlagendaten der Messung „AFB“

Hinweis: Die Konfiguration der InfluxDB erfolgt wie im gleichnamigen Abschnitt beschrieben und wird auf *localhost* betrieben.

InfluxDB

Auf einem Windows-System erfolgt das Ausführen einer InfluxDB mit Hilfe eines vorkompilierten Daemon per `influxd.exe`. Damit steht unter der Adresse `localhost:8086` die Datenbank-Web-Oberfläche zur Verfügung, welche nun zur Konfiguration verwendet wird. Für die avisierten Buckets benötigen wir (ein) TOKEN als Authentisierung.

Die Weboberfläche wird ebenfalls zur Anlage von Notebooks für flux-Abfragen verwendet.

InfluxDB-Client

Eine komfortablere Abfrage und Auswertung von InfluxDB-Zeitreihendaten erfolgt mittels des Python Moduls `influxdb_client`. Eine flux-Query wird der `query_api` des Client zur Verfügung gestellt (Abbildung 1) und die json-tables mittels funktionalem Mapper in eine csv-Datei geschrieben (Abbildung 2).

```
1  %%time
2
3  fname = "225c.flux"
4
5  with InfluxDBClient(url=url, token=token, org=org, debug=False, timeout=240000) as client:
6
7      query_api = client.query_api()
8      query = fqueries[fname]
9
10     tables = query_api.query(org=org, query=query)
11     write_(tables, fname=os.path.join(fpath,"%s.txt"%(fname)))
12
✓ 9.7s
```

Abbildung 1

```

28 def write_(tables,fname,verbose=0):
29     # load json result
30     output = json.loads(tables.to_json())
31
32     ls = "\n"
33     sep = "|"
34
35     if verbose: print(f"write {os.path.basename(fname)}")
36
37     with open(fname,"w") as f:
38         # header
39         columns = list(output[0].keys())
40         _ = [columns.remove(c) for c in ['result']]
41         rc = sep.join(c for c in columns)
42         v = f"{rc}{ls}"
43         f.write(v)
44         if verbose: print(v)
45         # values
46         for row in output:
47             rc = sep.join(str(row[c]) for c in columns)
48             v = f"{rc}{ls}"
49             f.write(v)
50             if verbose: print(v)
51
52     if verbose: print(f"wrote {len(output)} line(s) of {fname}")

```

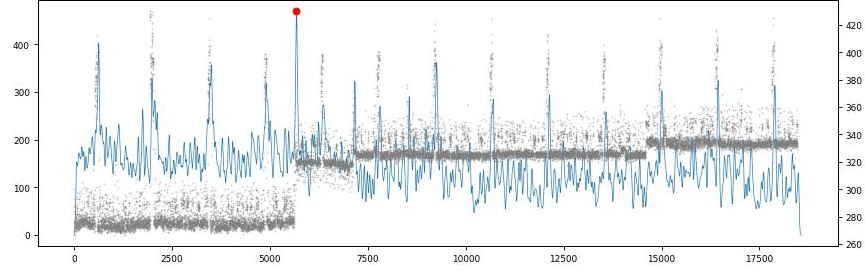
Abbildung 2

Die nachfolgende Aufgabenbearbeitung folgt dem Schema <aufgabennummer>.flux für die Query, <aufgabennummer>.flux.txt für Ergebnismenge.

Stromdaten

Bucket „power“, Aufgabe 2.2

	Ident	Bemerkung
1	221	inkl. „delay“ in Sek.
2	222	24 h Auflösung, Summe der „_value“ über alle Sensoren in „consumption“, für Gruppierung jeweils eigene <i>tables</i>
3	223	24 h Auflösung der „hour“ für jeden Raum „location“ in „consumption“
4	224	time join der „Device“ Queries „server1“ und „server2“ mit „_value“ als Summe; Reduktion der Zieldatenmenge über „aggregateWindow“ von „5m“; Hinweis: „None“ „_value“, entstanden durch die Aggregation-Windows, können in den Zieldaten entfernt werden
5a	225a	„server1“ Zeitreihe über „aggregateWindow“ von „10m“; Korrektur der „Ausreißer“ über Mittelwert; Anstiegsermittlung der Aggregation mittels „derivative“ und einer „duration“ von „10s“; „bulge“ > 0.2 weist auf emergentes Verhalten hin; visuelle Inspektion der 9 Zeitdaten und Auswahl Anhand einer Orientierung aus Abbildung 3 mit „2023-04-11T11:40:00+00:00“

		<p>sudden increase 2023-04-11T12:48:00+00:00</p>  <p>Abbildung 3</p>
5b	225b	Einsatz der Auswahl „2023-04-11T11:40:00+00:00“ aus 225a für „tevent“ mit „past“ und „before“ als Mittelwert
5c	225c	minimale Korrektur durch „map“ auf „Device“ „server1“
6	226	nach 225c; Auffinden von Korrekturen „corrected“ und verbleibenden Ausreißern „outlier“; Annahme „echte Ausreißer“ liegen zwischen 3 und 22 Uhr (tagsüber); In „server1“ nach dieser Maßgabe genau einen Ausreißer um „2023-04-13T11:58:46.866152+00:00“ gefunden

Anlagendaten

Bucket „factory“; Aufgabe 2.3; AFB

	Ident	
1	231	inkl. „delay“ in Sek.
2	232	Aggregation über „Assembly“ als „Signal“ „count“; „_all_“ für „assembly“ trägt Gesamtsumme
3	233	„_lorry“ als Nummer; „_count“ als Anzahl der Umläufe; insgesamt 4 Wagen im Set
4	234a	Jeder „lorry“ benötigt im Mittel „cycle_mean“ Zeit mit einer Standardabweichung in „cycle_stddev“
	234b	„_lorry“ mit „_batch“ Ladungen, die nicht leer („_batch=4“) sind

Bucket „factory2“; Aufgabe 2.3; DS_SS23_TrayStates_B

Unterordner 2		
	Ident	
1	231	inkl. „delay“ in Sek.
2	232	Aggregation über „Assembly“ als „Signal“ „count“; „_all_“ für „assembly“ trägt Gesamtsumme; (2 Assemblies)
3	233	„_lorry“ als Nummer; „_count“ als Anzahl der Umläufe – jeder Wagen 6 Umläufe; insgesamt 4 Wagen im Set
4	234a	Jeder „lorry“ benötigt im Mittel „cycle_mean“ Zeit mit einer Standardabweichung in „cycle_stddev“
	234b	„_lorry“ mit „_batch“ Ladungen, die nicht leer („_batch=4“) sind; „_loadings“ mit 1 gekennzeichnet, Abbildung 4

