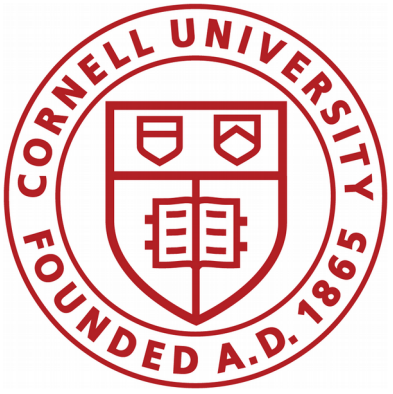


CBPM3 development

– data base rate test –

Antoine

CBPM meeting

March 27, 2020

Goal

The design for CBPM3 is:

- ✗ 10 Hz continuous read-out of about 120 CBPMs
- ✗ data stored in data base with a 2 week history

Want to demonstrate:

- ✗ ability to populate a data base at 10 Hz mimicking data from 120 CBPMs
- ✗ reliable continuous running for 2 weeks

Data base rate test

Setup:

- ✗ postgreSQL server installed on /nfs/ilc/sim3
- ✗ postgreSQL server running from desktop lnx6248
- ✗ one data base called test_cbpm3
- ✗ one table per CBPM, i.e. 120 tables (“instr0”, “instr1”… “instr119”)
- ✗ each row of each table has 5 columns: one timestamp and four button values

Populating code:

- ✗ open one connection to DB and leave it open
- ✗ create one cursor (do the execute, commit…) and re-use it indefinitely
- ✗ in an infinite (`while 1 == 1:`) loop:
  - ➢ generate unique timestamp for all the 120 CBPMs
  - ➢ serially populate tables (`for i in range(120) :`)
  - ➢ wait 0.05 second and start populating again (0.05 was roughly tweaked to eye-ball about 10 Hz average)

Populating code

```python
def populate(conn, cur):
    timestamp = datetime.datetime.now()

    for i in range(120):
        sql = "INSERT INTO instr" + str(i) + "(timestamp, top_in, bot_in, bot_out, top_out) VALUES(%s, %s, %s, %s, %s);"

        cur.execute(
            sql,
            (
                str(timestamp),
                random.randint(5000, 20000),
                random.randint(5000, 20000),
                random.randint(5000, 20000),
                random.randint(5000, 20000)
            )
        )

    conn.commit()


def start():
    conn = None
    try:
        conn = psycopg2.connect(host="127.0.0.1", port=5432, database="test_cbpm3", user="antoine", password="")
        cur = conn.cursor()
        while 1 == 1:
            populate(conn, cur)
            time.sleep(0.05)
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)

    if conn is not None:
        conn.close()
        print('Database connection closed.')


start()
```

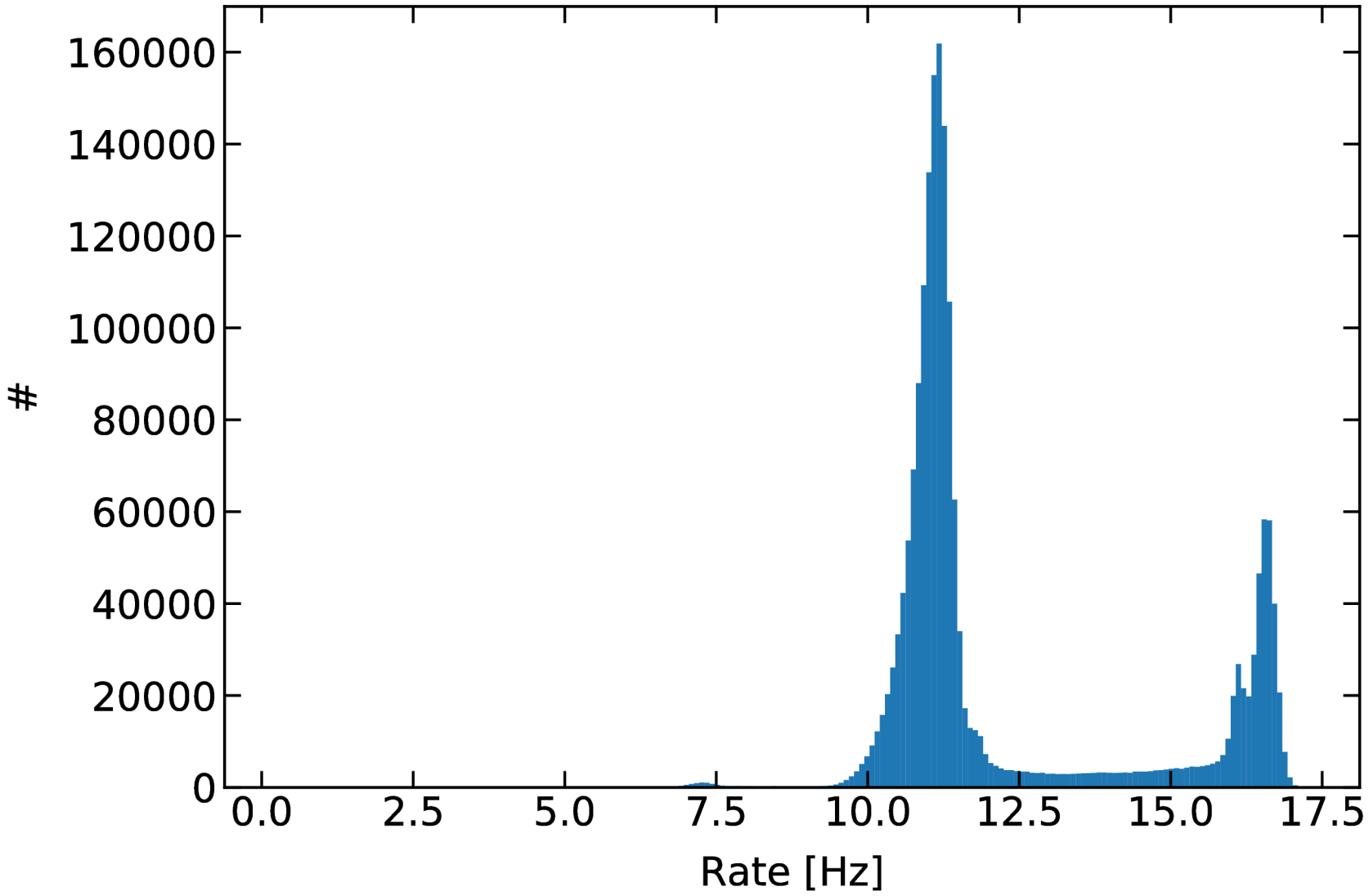
Example of one table in the data base

timestamp	top_in	bot_in	bot_out	top_out
2020-03-26 10:25:39.352366	5783	19818	11015	16269
2020-03-26 10:25:39.26489	16572	12288	11255	10283
2020-03-26 10:25:39.174568	5351	13926	7002	17351
2020-03-26 10:25:39.083338	7493	14962	12067	8526
2020-03-26 10:25:38.996268	19462	7479	18509	18401
2020-03-26 10:25:38.933941	10149	10030	15765	9475
2020-03-26 10:25:38.87157	16050	15977	15342	13349
2020-03-26 10:25:38.811682	14154	6223	6603	14370
2020-03-26 10:25:38.751498	6169	11992	13760	13875
2020-03-26 10:25:38.683063	11438	13509	7894	17750
2020-03-26 10:25:38.594163	8415	13108	13740	14143
2020-03-26 10:25:38.502603	14114	14994	18795	6192
2020-03-26 10:25:38.414299	5354	13391	14414	19307
2020-03-26 10:25:38.325104	15075	13520	8338	15689
2020-03-26 10:25:38.240226	14799	9347	12997	13132
2020-03-26 10:25:38.149437	10042	9646	15559	9310
2020-03-26 10:25:38.058211	10837	17303	17609	19370
2020-03-26 10:25:37.968018	12290	7077	12477	10747
2020-03-26 10:25:37.878672	14724	16168	17014	17095
2020-03-26 10:25:37.791534	6147	16967	8251	19524
2020-03-26 10:25:37.709076	18615	10667	13803	13289
2020-03-26 10:25:37.634555	15164	15534	12620	16605
2020-03-26 10:25:37.556595	17411	10941	12020	10420
2020-03-26 10:25:37.466434	11511	11579	18022	18274
2020-03-26 10:25:37.377055	8076	9561	16088	17176
2020-03-26 10:25:37.287734	16738	7670	14191	15279
2020-03-26 10:25:37.199572	6869	6466	18008	6394
2020-03-26 10:25:37.109127	12249	17092	17100	16020
2020-03-26 10:25:37.016535	8248	5423	10606	13706
2020-03-26 10:25:36.927494	15904	13442	9298	8578
2020-03-26 10:25:36.839783	12858	9657	10199	15470
2020-03-26 10:25:36.773006	11358	5870	14521	18295

2-week stress test

Code started Tuesday 3/25 at 2PM with the hope of running non stop for 2 weeks.

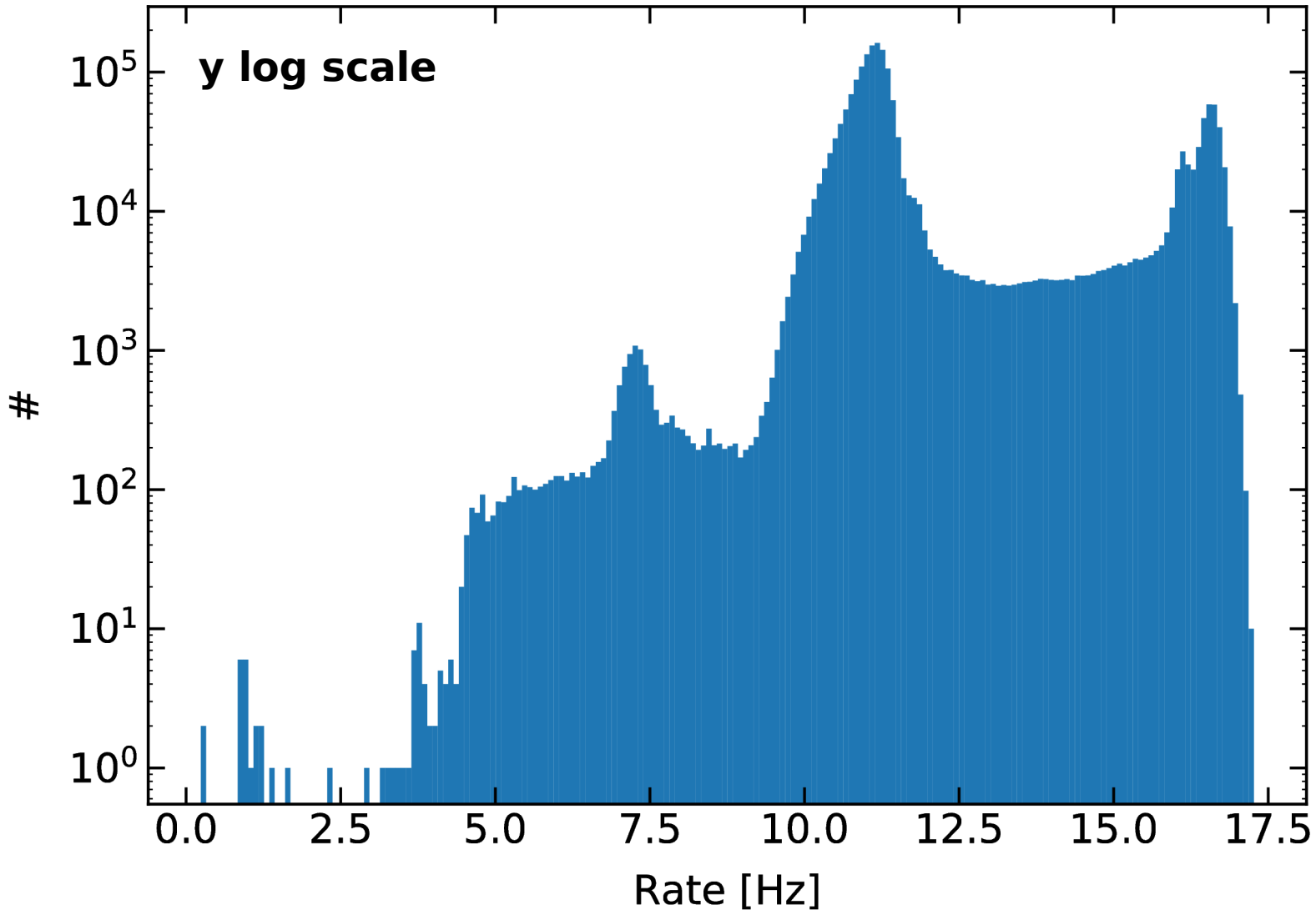
After 48 hours of running, average rate is a steady 12 Hz.



2-week stress test

Code started Tuesday 3/25 at 2PM with the hope of running non stop for 2 weeks.

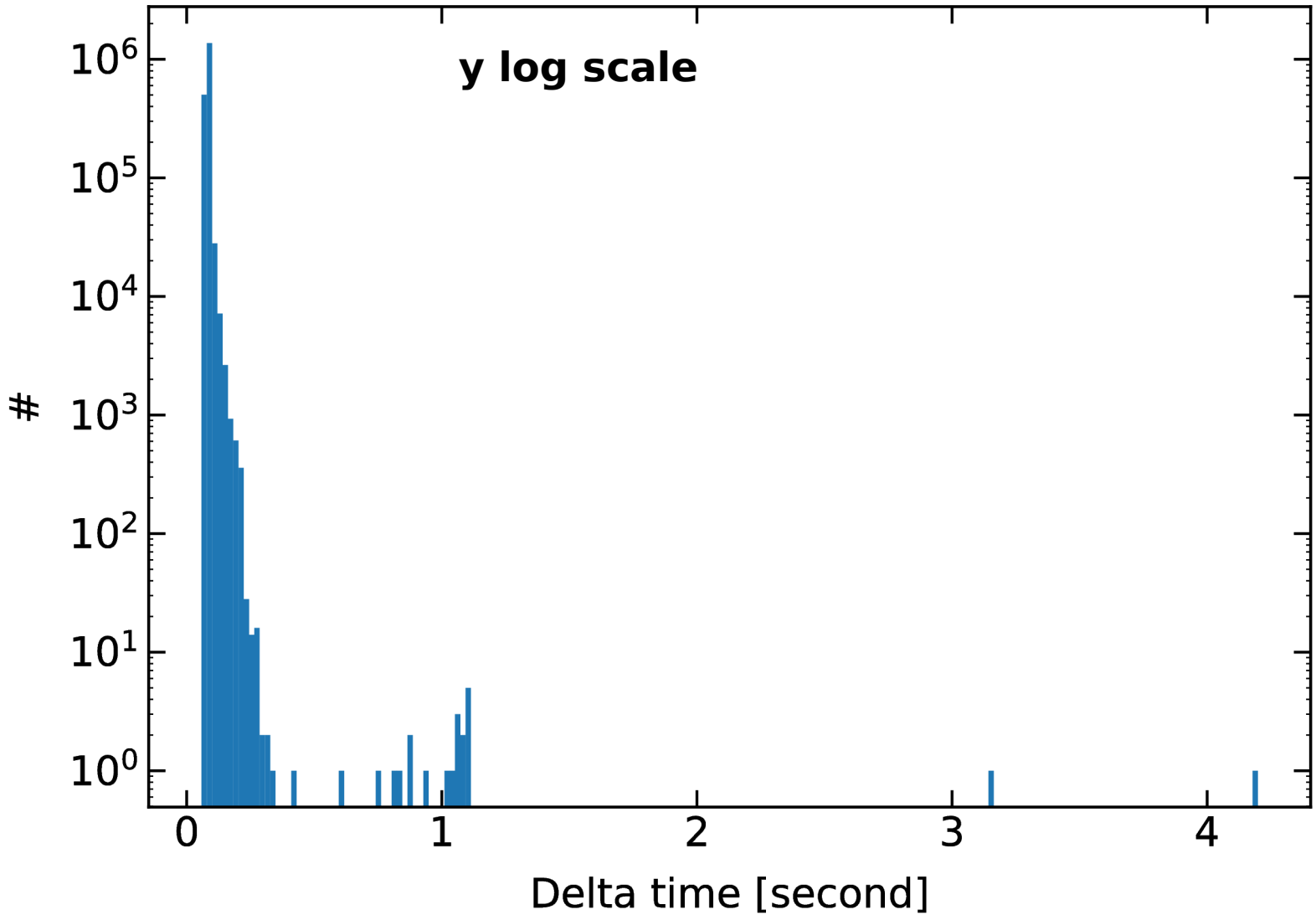
After 48 hours of running, average rate is a steady 12 Hz. **But with variations...**



2-week stress test

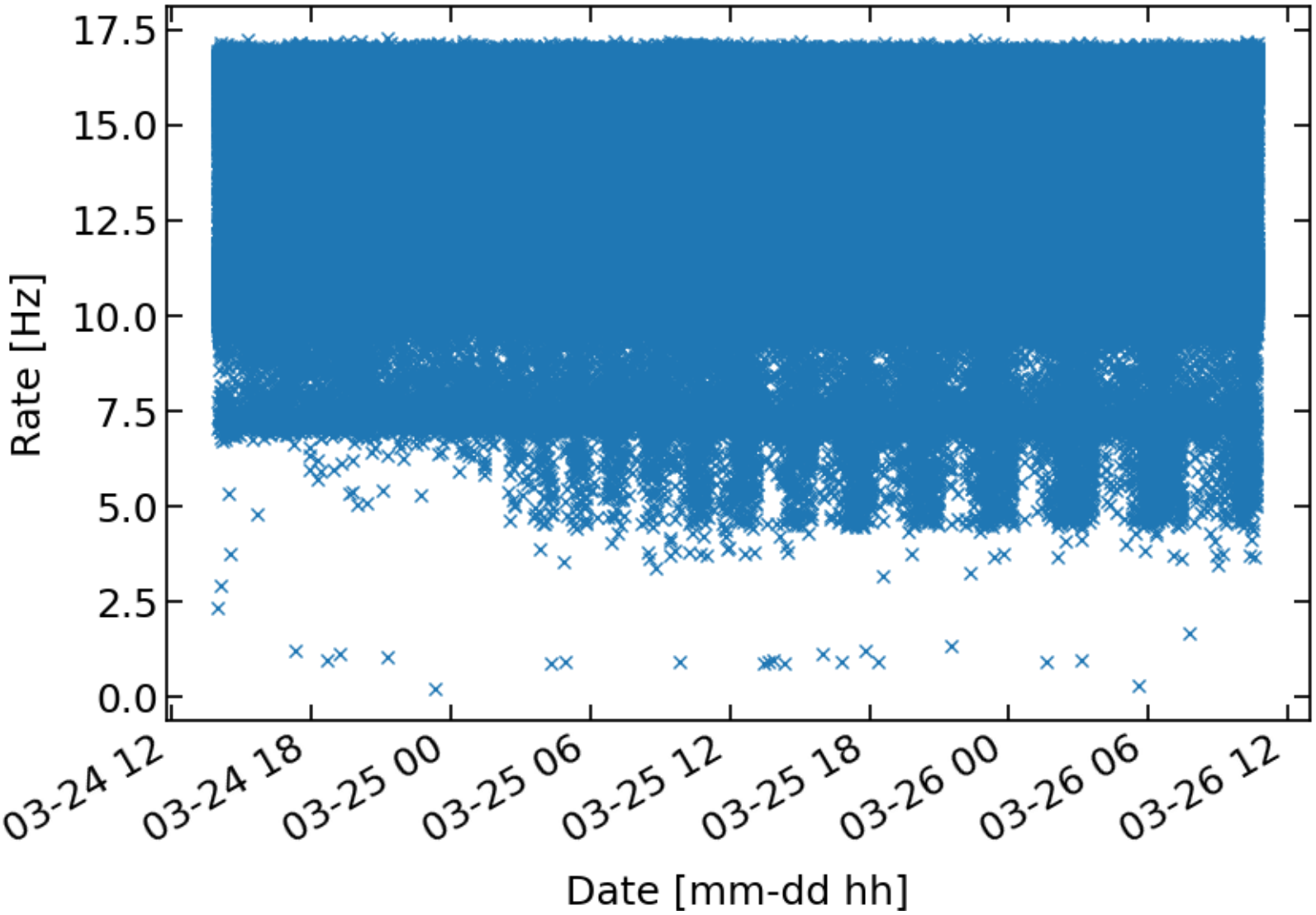
Code started Tuesday 3/25 at 2PM with the hope of running non stop for 2 weeks.

After 48 hours of running, average rate is a steady 12 Hz. **In the time domain.**



2-week stress test

There seems to be patterns in the rate history: caused by nfs? networking? data base itself? Plus: is the rate slowly degrading? We'll keep monitoring.



Data base size, data throughput

Each table row has:

- One timestamp → 1 x 8-byte data type (“timestamp without time zone”)
- 4 button values → 4 x 4-byte data type (“real”)

For 120 CBPMs at 12 Hz:

- expect about 2.1 MB/minute
- reality is closer to 4.3 MB/minute

Difference not understood. Is there a factor two overhead in storing the information?

Given 4.3 MB/minute → 87 GB for continuously running at 12 Hz for 2 weeks.

Outlook

First data base feasibility test successful so far! Things to do/think about:

- ✗ keep monitoring the situation for the remainder of the 2-week test
- ✗ experiment concurrent (async) vs serial data base population
- ✗ network monitoring/get in touch with IT group
- ✗ expand test to pretend data from 120 CBPMs are presented at 10 Hz
- ✗ read-out rate test: can we read fast enough from data base while writing to it
- ✗ maximum simultaneous read/write rate?
- ✗ create dummy on-line display to show orbit at 10 Hz

Additional materials