

Разработка каталога физических событий эксперимента SPD на ускорителе частиц NICA: Оптимизация загрузки и чтения данных

Исполнитель: **Будтуева Замира Алановна**
Научные руководители: **Прокошин Фёдор Валерьевич** (ОИЯИ),
Тваури Инга Васильевна(СОГУ)

Задачи

SPD EventIndex - это каталог физических событий, полученных с детектора SPD на коллайдере NICA или смоделированных для анализа и обработки данных.

EventIndex будет использоваться на детекторе SPD, с которого ожидается получение 30 миллиардов событий в год.

- 1.Необходима система способная справиться с потоком в десятки тысяч событий в секунду.**
- 2.Также важной задачей является оптимизация работы разрабатываемой системы.**

Модули для взаимодействия базы данных PostgreSQL и Python скрипта для чтения и записи событий

Асунсрпг

Aiopg

SQL-Alchemy

Pg800

Psycopg2

PyGreSQL

Py-postgresql

Таблицы базы данных

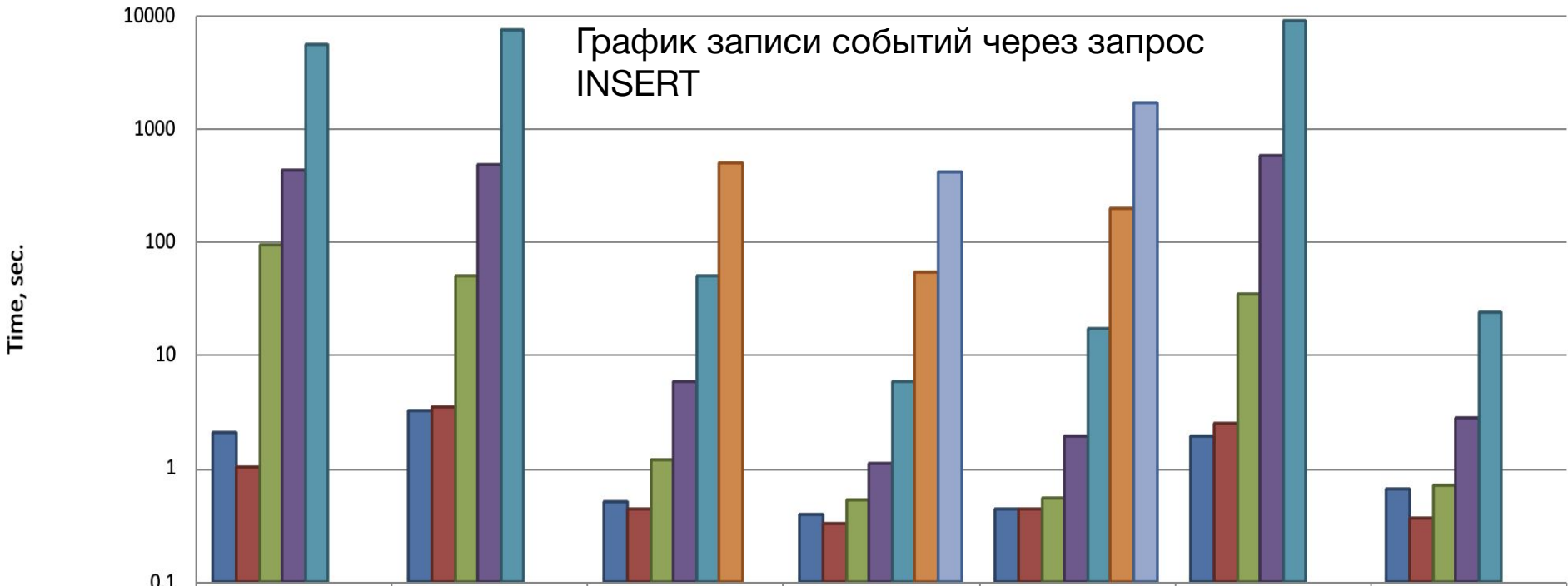
run_number	event_number	olf_result	dsid_raw	fuid_raw
280308061	12	21810	1	dfcdd203-a473-4c0c-909d-a35252512595
280308061	30	17705	1	dfcdd203-a473-4c0c-909d-a35252512595
280308061	43	19125	1	dfcdd203-a473-4c0c-909d-a35252512595

Рис. 1: Таблица events

dsid_raw	dsn_raw
1	data28_pp_9p4GeV.280308061.RAW.t0001
2	data28_pp_9p4GeV.280308123.RAW.t0001
3	data28_pp_9p4GeV.280308042.RAW.t0001

Рис. 2: Таблица datasets

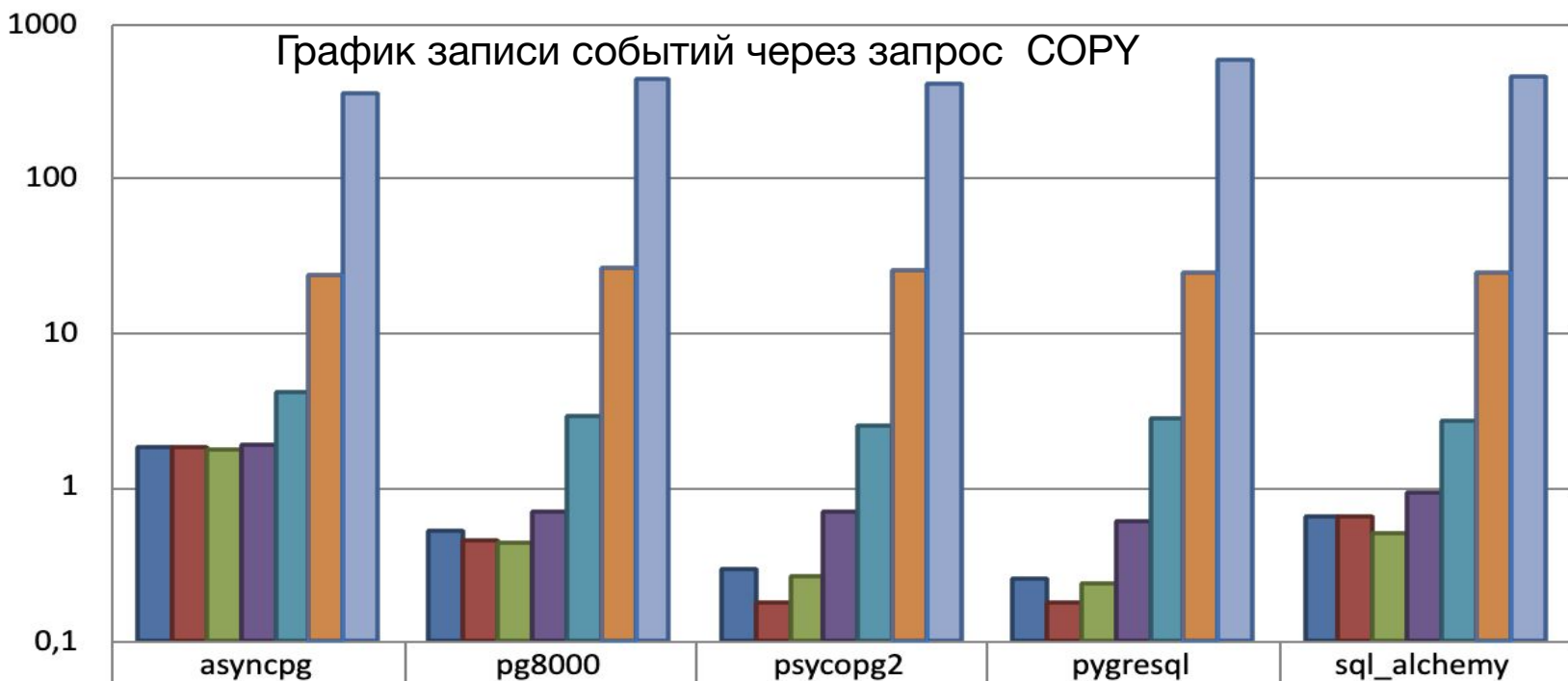
График записи событий через запрос INSERT



	aiopg	asyncpg	pg8000	psycopg2	pygresql	py-postgresql	sql_alchemy
10 events	2,05	3,21	0,50	0,40	0,44	1,93	0,66
100 events	1,02	3,45	0,45	0,33	0,44	2,47	0,36
1 000 events	93,47	49,84	1,17	0,52	0,55	34,84	0,70
10 000 events	432,35	470,17	5,86	1,11	1,89	581,96	2,77
100 000 events	5484,15	7296,31	50,53	5,87	17,18	9037,89	24,24
1 000 000 events			500,55	54,21	194,08		
10 000 000 events				415,95	1698,68		

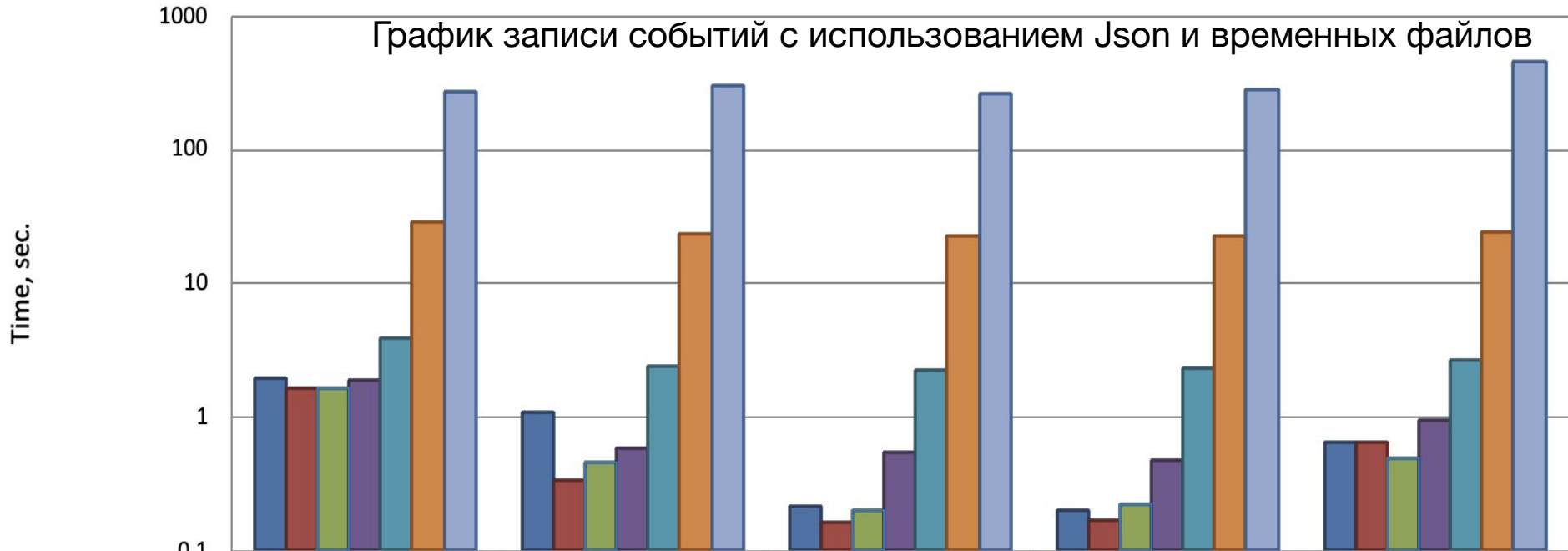
График записи событий через запрос COPY

Time, sec.



	asyncpg	pg8000	psycopg2	pygresql	sqlalchemy
10 events	1,79	0,52	0,29	0,25	0,64
100 events	1,79	0,44	0,18	0,18	0,65
1 000 events	1,74	0,44	0,27	0,24	0,49
10 000 events	1,89	0,70	0,69	0,61	0,93
100 000 events	4,08	2,85	2,52	2,73	2,69
1 000 000 events	23,44	25,87	24,92	24,13	24,33
10 000 000 events	353,36	441,02	412,77	574,04	449,64

График записи событий с использованием Json и временных файлов



	asyncpg	pg8000	psycopg2	pygresql	sqlalchemy
10 events	1,96	1,10	0,21	0,20	0,64
100 events	1,64	0,34	0,16	0,17	0,65
1 000 events	1,66	0,45	0,20	0,22	0,49
10 000 events	1,86	0,58	0,55	0,48	0,93
100 000 events	3,84	2,42	2,26	2,29	2,69
1 000 000 events	28,74	23,22	22,82	22,83	24,01
10 000 000 events	266,47	298,59	262,14	276,97	449,64

График записи событий с использованием только временных файлов

Time, sec.

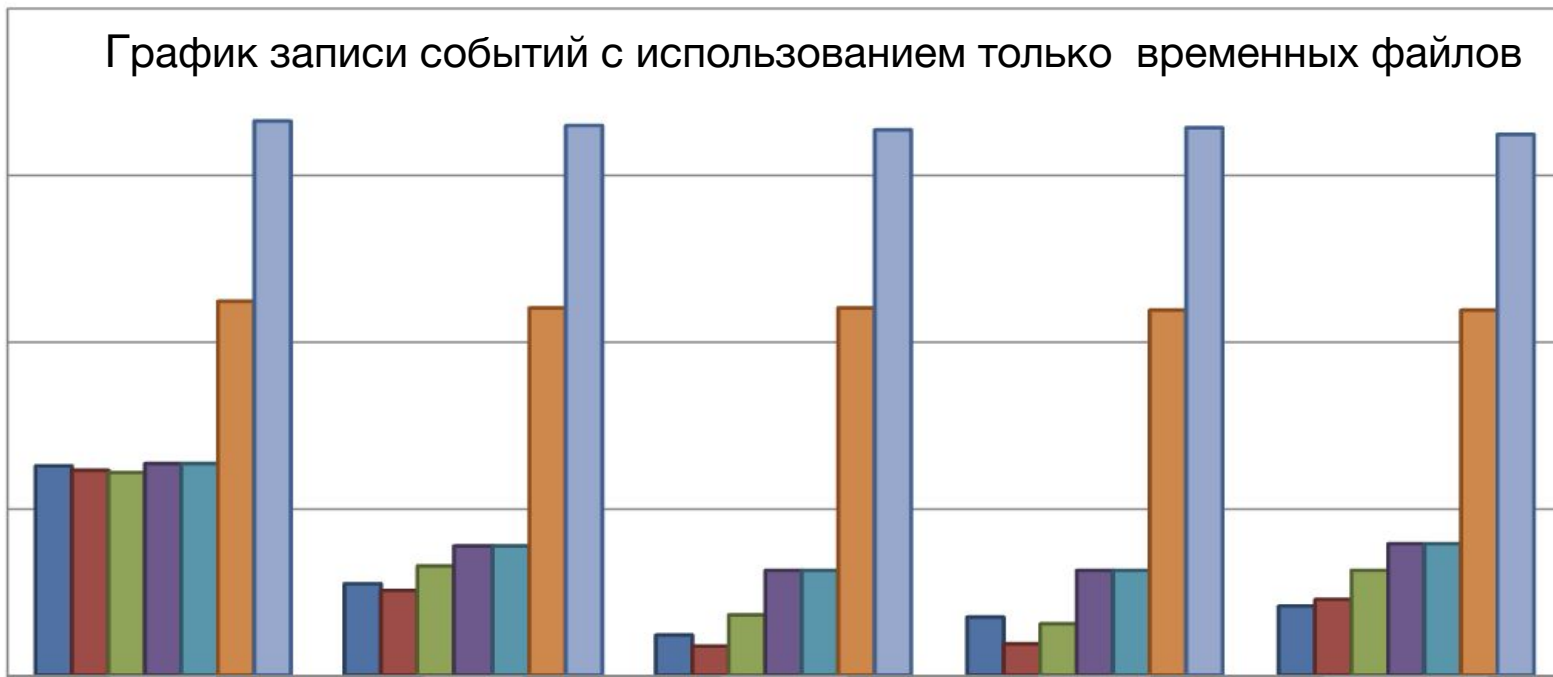
1000

100

10

1

0,1



asyncpg

pg8000

psycopg2

pygresql

sqlalchemy

10 events

1,79

0,35

0,18

0,22

0,26

100 events

1,69

0,32

0,15

0,15

0,28

1 000 events

1,65

0,44

0,23

0,21

0,42

10 000 events

1,86

0,59

0,43

0,42

0,62

100 000 events

1,86

0,59

0,43

0,42

0,62

1 000 000 events

17,13

15,83

15,59

15,38

15,21

10 000 000 events

206,40

195,27

186,57

186,86

174,13

Результаты профилирования для модуля SQL-Alchemy

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.164	0.164	0.164	0.164	store_events.py:142(<listcomp>)
1	18.948	18.948	182.055	182.055	store_events.py:31(main)
123724	0.130	0.000	0.130	0.000	{built-in method _codecs.utf_8_decode}
50	9.047	0.181	152.693	3.054	eipsyco.py:994(store_raw_event)
10011253/10010694	0.971	0.000	0.971	0.000	{built-in method builtins.len}
53	0.865	0.016	0.870	0.016	{built-in method psycopg2._psycopg2._connect}
10013250	1.226	0.000	1.226	0.000	{method 'append' of 'list' objects}
52	7.219	0.139	7.219	0.139	{method 'commit' of 'psycopg2.extensions.connection' objects}
51	119.633	2.346	119.633	2.346	{method 'copy_expert' of 'psycopg2.extensions.cursor' objects}
10000022	1.726	0.000	1.726	0.000	{method 'encode' of 'str' objects}
10005264	11.960	0.000	11.960	0.000	{method 'join' of 'str' objects}
1000	1.695	0.002	1.906	0.002	{method 'readlines' of '_io._IOBase' objects}
6	1.204	0.201	183.259	30.543	{method 'run' of 'Context' objects}
10000901	4.284	0.000	4.284	0.000	{method 'split' of 'str' objects}
10004393	1.578	0.000	1.578	0.000	{method 'strip' of 'str' objects}
10000000	1.366	0.000	1.366	0.000	{method 'write' of '_io.BytesIO' objects}

Заключение

Протестированные разные методы оптимизации кода такие, как:

- использование массовой загрузки
- применение разных синхронных и асинхронных модулей
- замена запроса INSERT на запрос COPY
- использование временных файлов
- профилирование.

→ Чтение и запись данных у различных модулей занимала около 2 часов на 100 тысяч событий.


→ На данный момент для выполнения тех же процессов уходит около 3 минут на 10 миллионов событий.

→

При этом можно выделить модуль SQL-Alchemy, который показал лучшие результаты (2,5 минуты на 10 миллионов событий).

Дальнейшие задачи

- Интеграция модуля загрузки данных с системой управления потоками данных.
- Адаптация API чтения физических данных для их индексации.
- Создание соответствующего программного модуля для использования в системе распределенных вычислений ranDA
- Оптимизация схемы БД для уменьшения размера и повышения производительности
- Оптимизация загрузки данных в рамках полного цикла и на высокопроизводительном оборудовании



Благодарю за внимание

