COMPRESS UNCOMPRESS UNCOMPRESS _LENGHT WEIGHT_STRING性能对比测试

**************************** COMPRESS **

一、 MySQL

1、 不查表

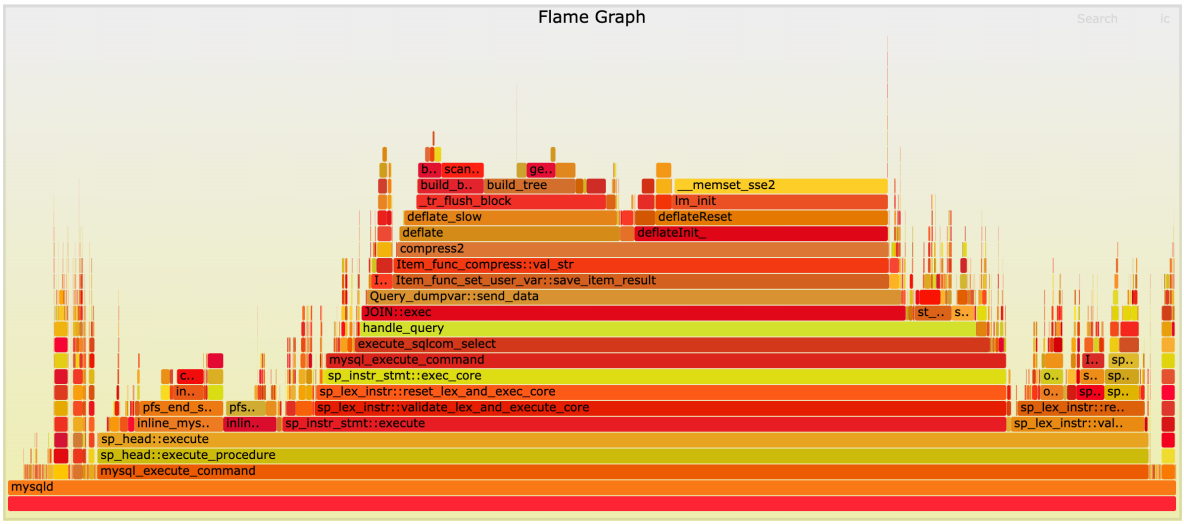
a) 测试语句

```
DROP PROCEDURE test_compress;
DELIMITER //
CREATE PROCEDURE test_compress()
BEGIN
   DECLARE i INT;
   DECLARE start TIME;
   DECLARE end TIME;

   SET i = 0;
   SET start = NOW();

   WHILE i < 2000000 DO
      SELECT COMPRESS('Test String. ') INTO @dummy;
      SET i = i + 1;
   END WHILE;

   SET end = NOW();
   SELECT TIMEDIFF(end, start) AS total_time;
END //
DELIMITER ;

CALL test_compress();
```

b) 测试结果

26s

c) 火焰图



2、 查表

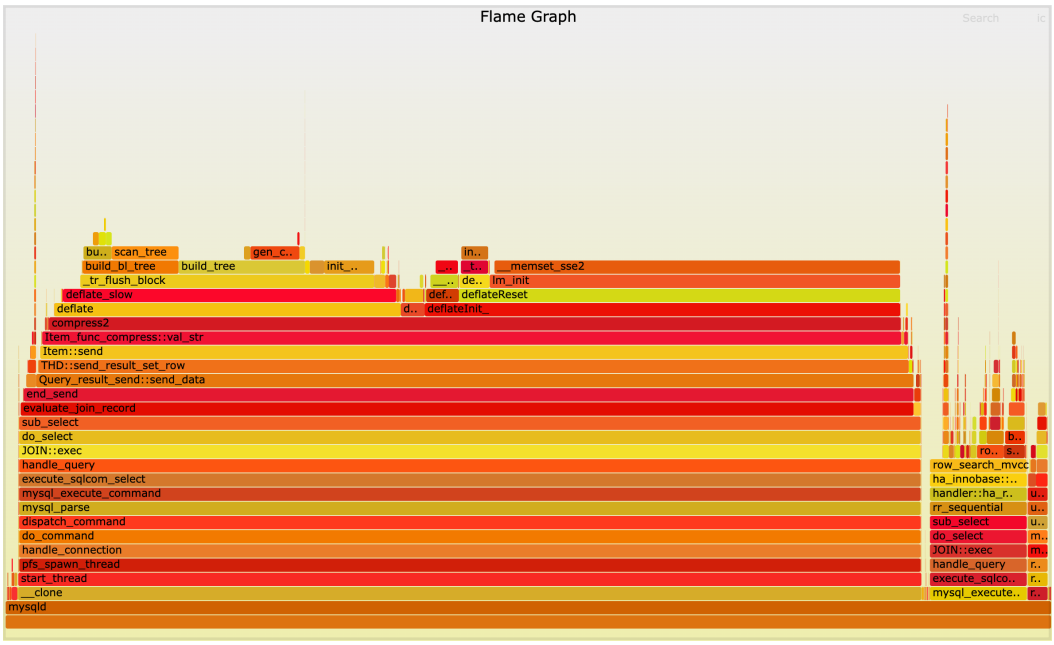
a) 测试语句

SELECT COMPRESS(y) from test_table_string;

b) 测试结果

54.80s

c) 火焰图



二、 openGauss

1、 不查表

a) 测试语句

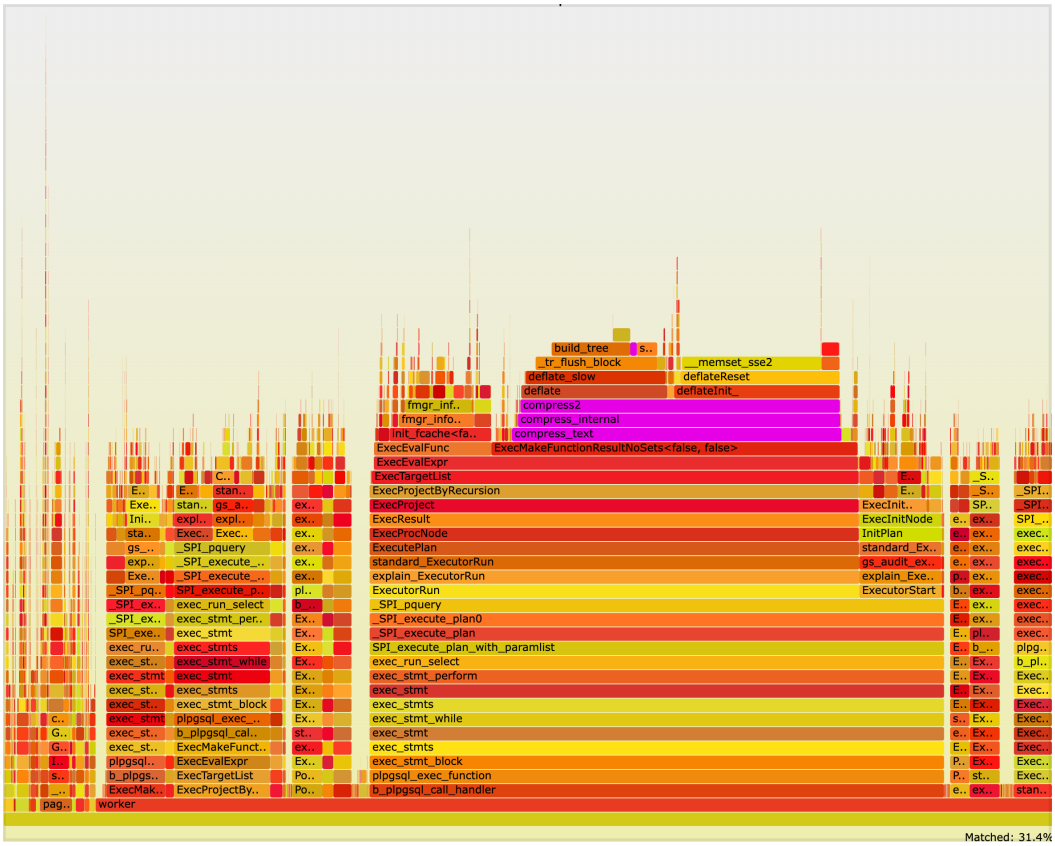
```
DROP FUNCTION test_compress();
CREATE OR REPLACE FUNCTION test_compress()
RETURNS interval AS
$$
DECLARE
```

```
        i INT;
        start_time TIMESTAMP;
        end_time TIMESTAMP;
        total INTERVAL;
    BEGIN
        i := 0;
        start_time := statement_timestamp();
        RAISE NOTICE 'start time: %', start_time;

        WHILE i < 2000000 LOOP
            PERFORM COMPRESS('Test String. ');
            i := i + 1;
        END LOOP;

        end_time := clock_timestamp();
        RAISE NOTICE 'end time: %', end_time;
        total :=  end_time - start_time;
                RAISE NOTICE 'total time: % ms', EXTRACT(MILLISECOND from
end_time-start_time);
        RETURN total;
    END
    $$
    LANGUAGE 'plpgsql';

    select test_compress();
```

b) 测试结果

34.5s，与 MySQL 比较下降 32%。

c) 火焰图



2、 查表

a) 测试语句

```
select compress(y) from test_table_string;
```

b) 测试结果

59.40s,与 MySQL 比较性能下降 8.4%。

c) 火焰图

********** UNCOMPRESS **********

一、 MySQL

1、 不查表

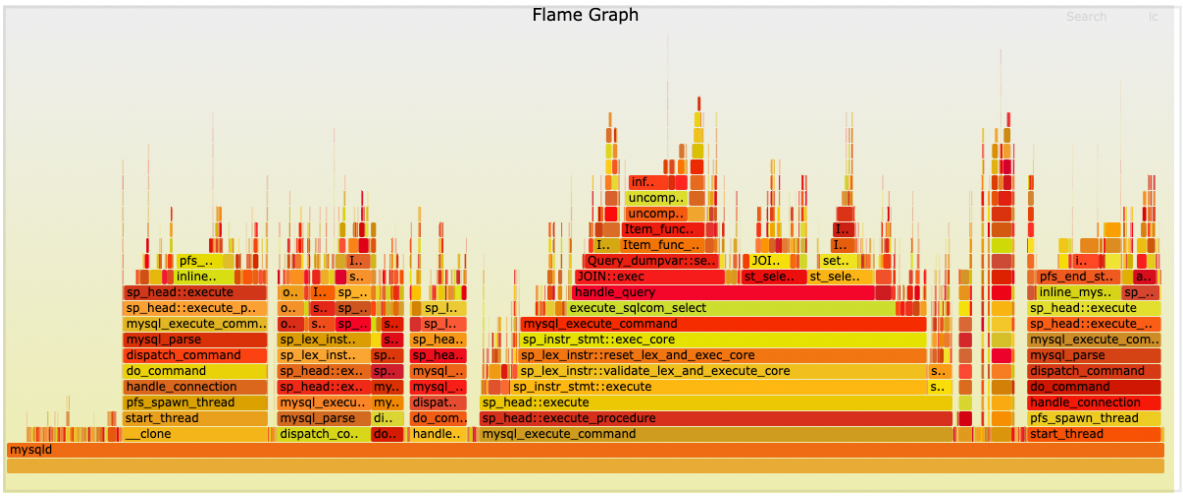
a) 测试语句

```
DROP PROCEDURE test_uncompress();

DELIMITER //
CREATE PROCEDURE test_uncompress()
BEGIN
  DECLARE i INT;
  DECLARE start TIME;
  DECLARE end TIME;

  SET i = 0;
  SET start = NOW();
  SET @var = COMPRESS('Test String. ');

  WHILE i < 2000000 DO
    SELECT UNCOMPRESS(@var) INTO @dummy;
    SET i = i + 1;
  END WHILE;

  SET end = NOW();
  SELECT TIMEDIFF(end, start) AS total_time;
END //
DELIMITER ;

CALL test_uncompress();
```

b) 测试结果

20s

c) 火焰图



2、 查表

a) 测试语句

SELECT UNCOMPRESS(z) from test_table_string;

b) 测试结果

12.13s

c) 火焰图

二、 openGauss

1、 不查表

a) 测试语句

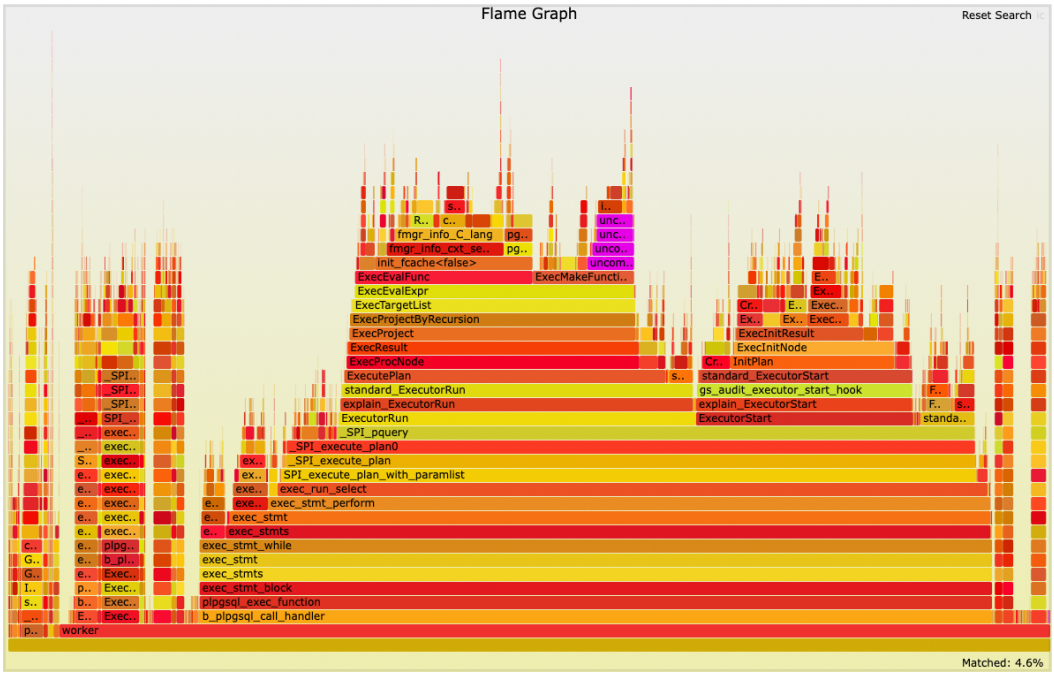
```
DROP FUNCTION test_uncompress();
CREATE OR REPLACE FUNCTION test_uncompress()
RETURNS interval AS
$$
DECLARE
   i INT;
   start_time TIMESTAMP;
   end_time TIMESTAMP;
   total INTERVAL;
   var bytea;
BEGIN
   i := 0;
   var := COMPRESS('Test String. ');
   start_time := statement_timestamp();
   RAISE NOTICE 'start time: %', start_time;

   WHILE i < 2000000 LOOP
      PERFORM UNCOMPRESS(var);
      i := i + 1;
   END LOOP;
```

```
        end_time := clock_timestamp();
        RAISE NOTICE 'end time: %', end_time;
        total :=  end_time - start_time;
                RAISE NOTICE 'total time: % ms', EXTRACT(MILLISECOND from
end_time-start_time);
        RETURN total;
    END
    $$
    LANGUAGE 'plpgsql';

    select test_uncompress();
```

b) 测试结果

21s,与 MySQL 性能接近。

c) 火焰图



2、 查表

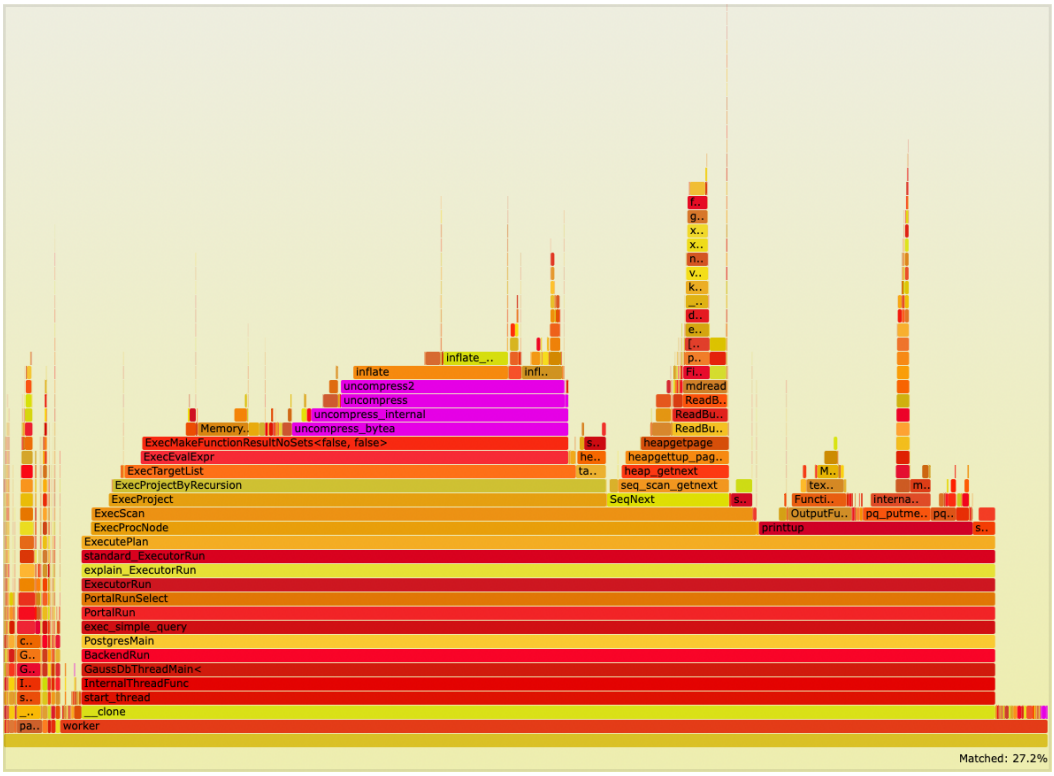
a) 测试语句

```
select UNCOMPRESS(z) from test_table_string;
```

b) 测试结果

9.47s，性能比 MySQL 提升 21%。

c) 火焰图



********** UNCOMPRESSED_LENGTH **********

一、 MySQL

1、 不查表

a) 测试语句

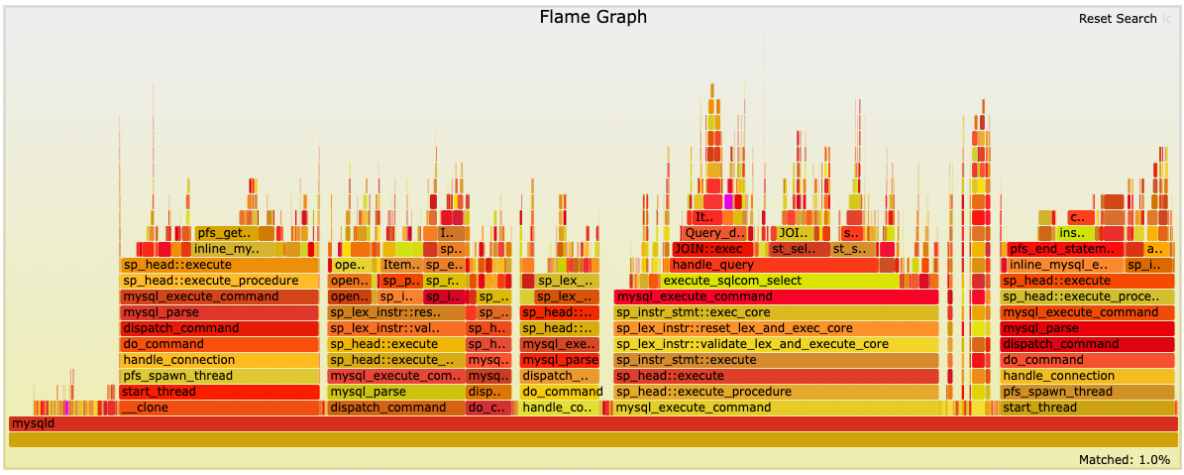
```
DROP PROCEDURE test_compress_len;
DELIMITER //
CREATE PROCEDURE test_compress_len()
BEGIN
  DECLARE i INT;
  DECLARE start TIME;
  DECLARE end TIME;

  SET i = 0;
  SET @var = COMPRESS('Test String. ');
  SET start = NOW();

  WHILE i < 2000000 DO
    SELECT UNCOMPRESSED_LENGTH(@var) INTO @dummy;
    SET i = i + 1;
  END WHILE;

  SET end = NOW();
  SELECT TIMEDIFF(end, start) AS total_time;
```

```
END //
DELIMITER ;

CALL test_compress_len();
```

b) 测试结果

17.66s

c) 火焰图



2、 查表

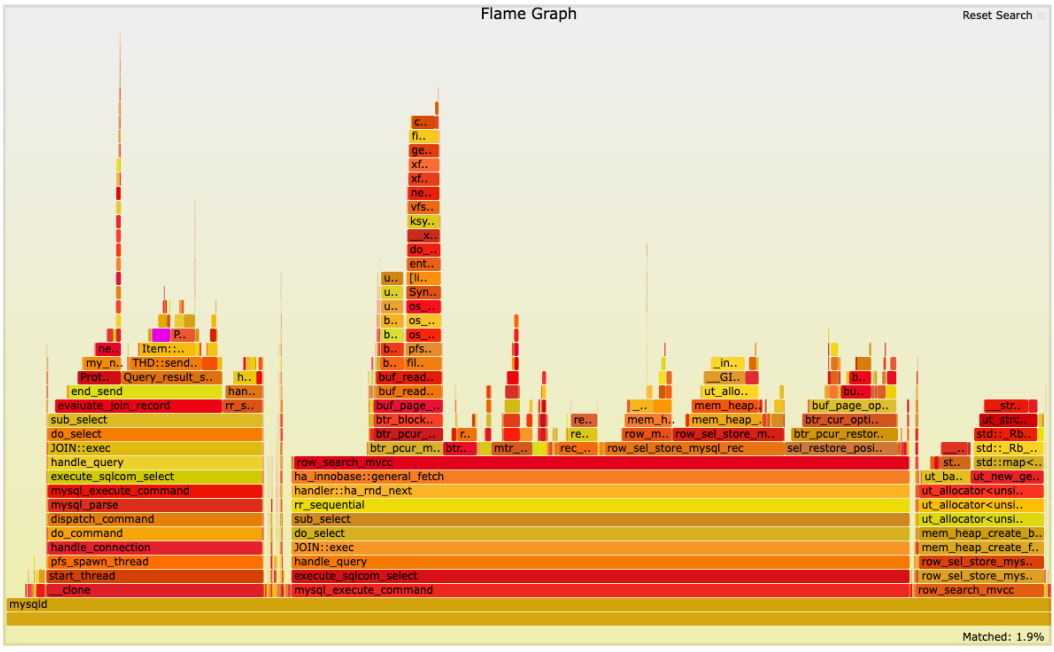
a) 测试语句

```
SELECT UNCOMPRESSED_LENGTH(z) from test_table_string;
```

b) 测试结果

7.76s

c) 火焰图



二、 openGauss

1、 不查表

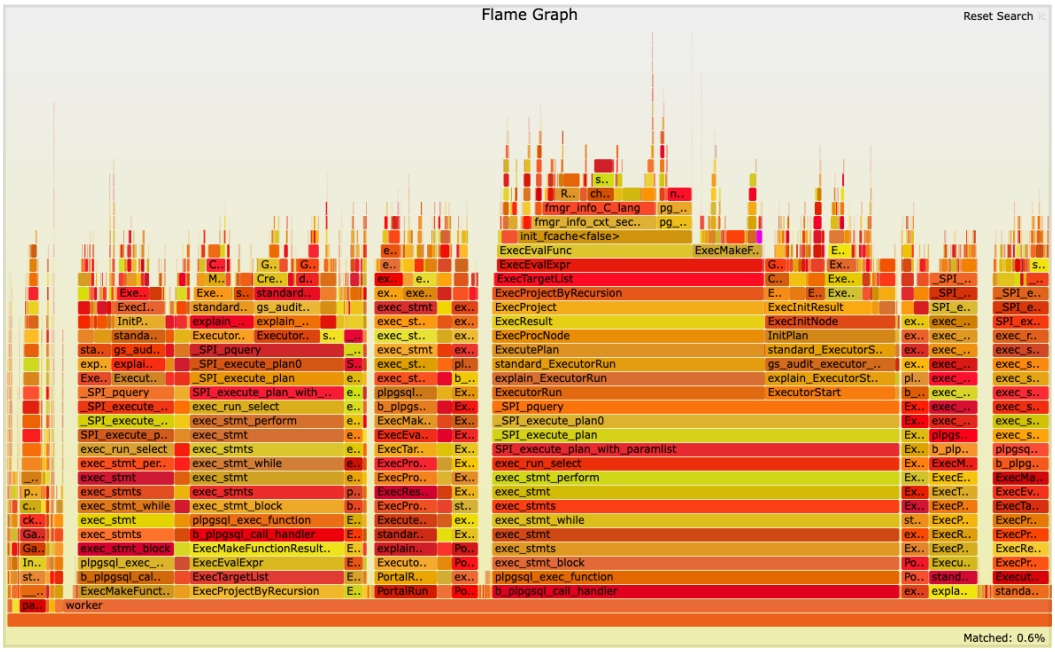
a) 测试语句

```
DROP FUNCTION test_uncompress_len;
CREATE OR REPLACE FUNCTION test_uncompress_len()
RETURNS interval AS
$$
DECLARE
   i INT;
   start_time TIMESTAMP;
   end_time TIMESTAMP;
   total INTERVAL;
   var bytea;
BEGIN
   i := 0;
   var := COMPRESS('Test String. ');
   start_time := statement_timestamp();
   RAISE NOTICE 'start time: %', start_time;

   WHILE i < 2000000 LOOP
      PERFORM UNCOMPRESSED_LENGTH(var);
      i := i + 1;
   END LOOP;

   end_time := clock_timestamp();
   RAISE NOTICE 'end time: %', end_time;
   total :=  end_time - start_time;
            RAISE NOTICE 'total time: % ms', EXTRACT(MILLISECOND from
end_time-start_time);
   RETURN total;
END
$$
LANGUAGE 'plpgsql';

select test_uncompress_len();
```

b) 测试结果

19.69s,与 MySQL 相比性能下降 11%。

c) 火焰图



2、 查表

a) 测试语句

select UNCOMPRESSED_LENGTH(z) from test_table_string;

b) 测试结果

6.13s,性能较 MySQL 提升 21%。

c) 火焰图

* * * \

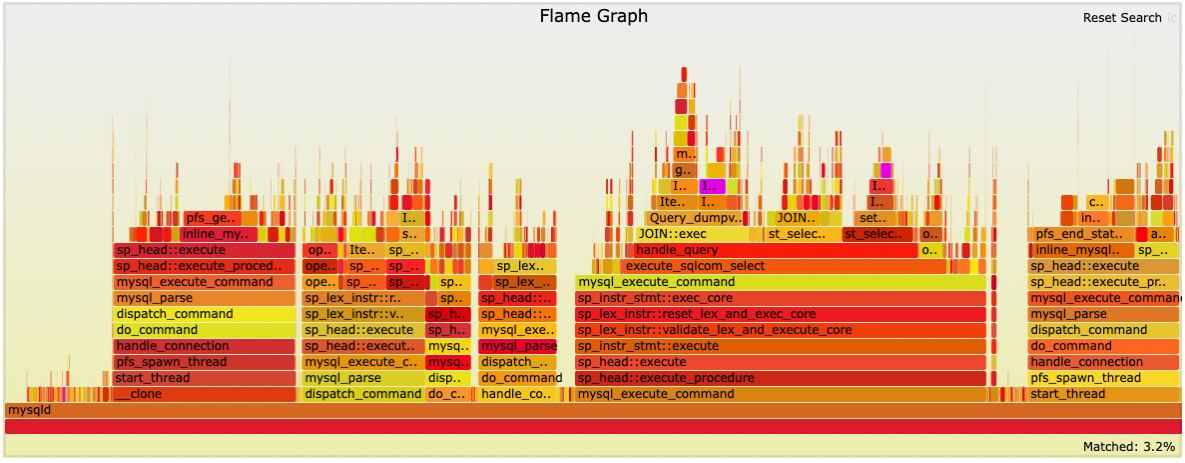
a) 测试语句

```
DROP PROCEDURE test_weight_string;
DELIMITER //
CREATE PROCEDURE test_weight_string()
BEGIN
  DECLARE i INT;
  DECLARE start TIME;
  DECLARE end TIME;

  SET i = 0;
  SET start = NOW();

  WHILE i < 2000000 DO
    SELECT WEIGHT_STRING('Test String. ') INTO @dummy;
    SET i = i + 1;
  END WHILE;

  SET end = NOW();
  SELECT TIMEDIFF(end, start) AS total_time;
END //
DELIMITER ;

CALL test_weight_string();
```

b) 测试结果

17.84s

c) 火焰图



2、 查表

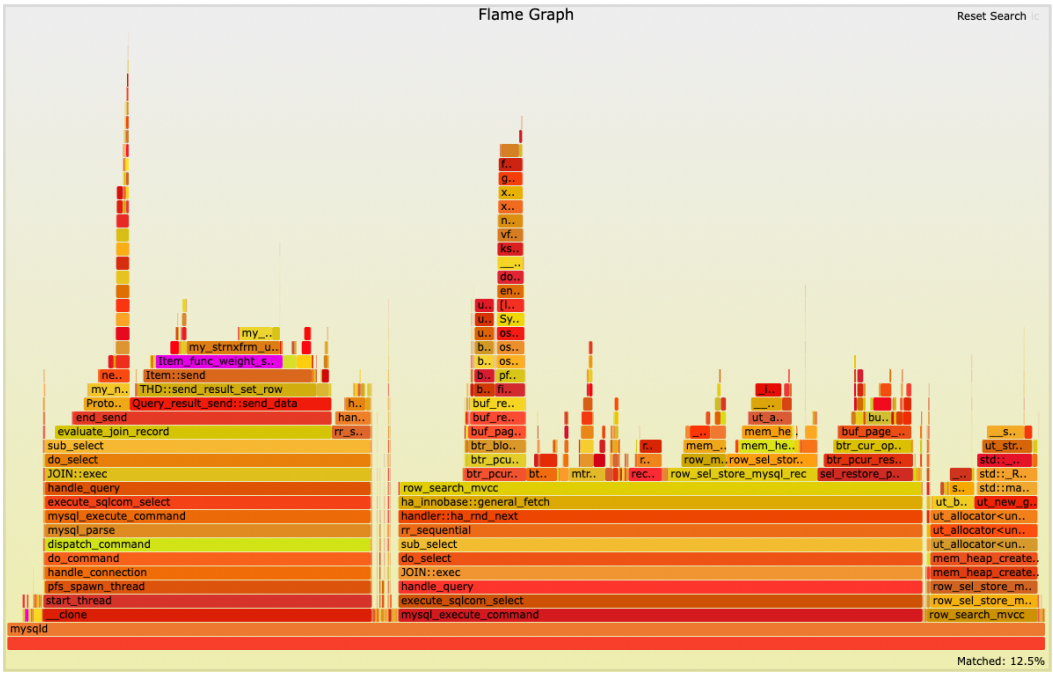
a) 测试语句

```
SELECT weight_string(y) FROM test_table_string;
```

b) 测试结果

9.03s

c) 火焰图



二、 openGauss

1、 不查表

a) 测试语句

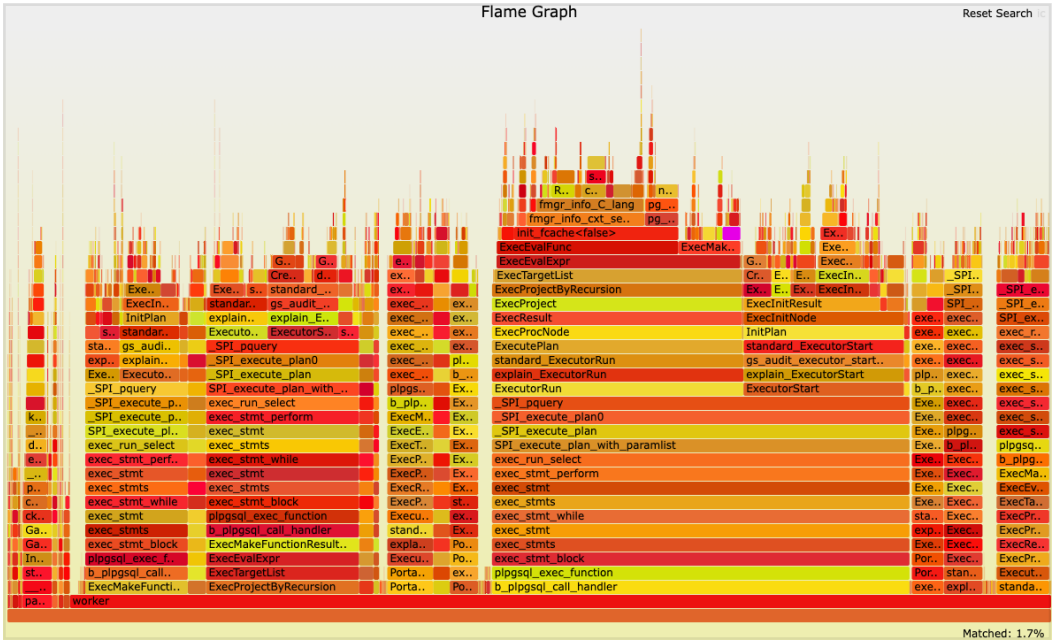
```
DROP FUNCTION test_weightstring();
CREATE OR REPLACE FUNCTION test_weightstring()
RETURNS interval AS
$$
DECLARE
   i INT;
   start_time TIMESTAMP;
   end_time TIMESTAMP;
   total INTERVAL;
BEGIN
   i := 0;
   start_time := statement_timestamp();
   RAISE NOTICE 'start time: %', start_time;

   WHILE i < 2000000 LOOP
      PERFORM WEIGHT_STRING('Test String. ');
      i := i + 1;
   END LOOP;

   end_time := clock_timestamp();
   RAISE NOTICE 'end time: %', end_time;
   total :=  end_time - start_time;
            RAISE NOTICE 'total time: % ms', EXTRACT(MILLISECOND from
end_time-start_time);
```

```
    RETURN total;
END
$$
LANGUAGE 'plpgsql';

select test_weightstring();
```

b) 测试结果

20.14s,相较 MySQL 性能下降 13%。

c) 火焰图



2、 查表

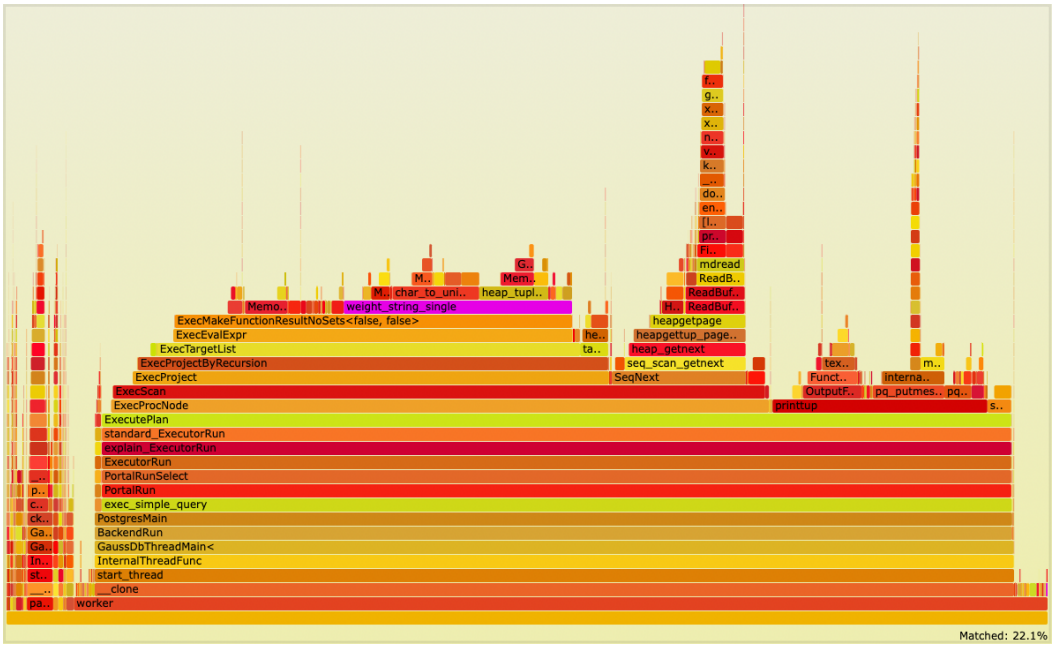
a) 测试语句

```
SELECT weight_string(y) FROM test_table_string;
```

b) 测试结果

8.77s,相较 MySQL 性能提升 3%。

c) 火焰图


f..
g..
x..
x..
n..
v..
k..
_..
do..
en..
[..
pr..
Fl..
mdread
ReadB..
ReadBuf..
ReadBuf..
heapgetpage
heapgettup_page..
heap_getnext
seq_scan_getnext
SeqNext
tex..
Funct..
OutputF..
printtup
m..
interna..
pq_putmes..
pq..
s..
G..
Mem..
M..
M..
char_to_uni..
heap_tupl..
H..
Memo..
weight_string_single
ExecMakeFunctionResultNoSets<false, false>
ExecEvalExpr
he..
ExecTargetList
ta..
ExecProjectByRecursion
ExecProject
ExecScan
ExecProcNode
ExecutePlan
standard_ExecutorRun
explain_ExecutorRun
ExecutorRun
PortalRunSelect
PortalRun
exec_simple_query
PostgresMain
BackendRun
GaussDbThreadMain<
InternalThreadFunc
start_thread
__clone
worker
_..
p..
c..
ck..
Ga..
Ga..
In..
st..
__..
pa..
Matched: 22.1%