shared_blks_hit_distinct
and how we can make it work

lukas@pganalyze.com

The Problem:

shared_blks_hit double counts

Test Setup:

```
CREATE TABLE a(id int);
CREATE TABLE b(id int, a_id int);

INSERT INTO a SELECT * FROM generate_series(0, 1_000_000);
INSERT INTO b SELECT a, a FROM generate_series(0, 1_000_000) x(a);

CREATE INDEX ON a(id);
CREATE INDEX ON b(id);
CREATE INDEX ON b(a_id);
VACUUM ANALYZE a;
VACUUM ANALYZE b;

SET max_parallel_workers_per_gather = 0;
```

Warm cache:

```
EXPLAIN ANALYZE
SELECT * FROM a JOIN b ON (a.id = b.a_id)
WHERE b.id BETWEEN 500 AND 5000;

Nested Loop  (...)
Buffers: shared hit=13540
->  Index Scan using b_id_idx on b  (...)
      Index Cond: ((id >= 500) AND (id <= 5000))
      Index Searches: 1
      Buffers: shared hit=36
->  Index Only Scan using a_id_idx on a  (...)
      Index Cond: (id = b.a_id)
      Heap Fetches: 0
      Index Searches: 4501
      Buffers: shared hit=13504
```

After a restart:

```
EXPLAIN ANALYZE
SELECT * FROM a JOIN b ON (a.id = b.a_id)
WHERE b.id BETWEEN 500 AND 5000;

Nested Loop  (...)
Buffers: shared hit=13493 read=47
->  Index Scan using b_id_idx on b  (...)
      Index Cond: ((id >= 500) AND (id <= 5000))
      Index Searches: 1
      Buffers: shared hit=2 read=34
->  Index Only Scan using a_id_idx on a  (...)
      Index Cond: (id = b.a_id)
      Heap Fetches: 0
      Index Searches: 4501
      Buffers: shared hit=13491 read=13
```

Storing all distinct buffer IDs would be too expensive.

HyperLogLog (HLL) to the rescue!

Existing structure does cumulative counts, hard to fit HLL structure in

```
typedef struct BufferUsage
{
    int64       shared_blks_hit;/* # of shared buffer hits */
    int64       shared_blks_read;
    int64       shared_blks_dirtied;
    int64       shared_blks_written;
    ...
} BufferUsage;

extern PGDLLIMPORT BufferUsage pgBufferUsage;
```

Buffer Usage counting is slow
because of diffing in
InstrStartNode/InstrStopNode

```
- /* dst += add - sub */
- void
- BufferUsageAccumDiff(BufferUsage *dst,
-                                     const BufferUsage *add,
-                                     const BufferUsage *sub)
- {
-         dst->shared_blks_hit += add->shared_blks_hit - sub->shared_blks_hit;
-         dst->shared_blks_read += add->shared_blks_read - sub->shared_blks_read;
-         dst->shared_blks_dirtied += add->shared_blks_dirtied - sub->shared_blks_dirtied;
-         dst->shared_blks_written += add->shared_blks_written - sub->shared_blks_written;
-         dst->local_blks_hit += add->local_blks_hit - sub->local_blks_hit;
-         dst->local_blks_read += add->local_blks_read - sub->local_blks_read;
-         dst->local_blks_dirtied += add->local_blks_dirtied - sub->local_blks_dirtied;
-         dst->local_blks_written += add->local_blks_written - sub->local_blks_written;
-         dst->temp_blks_read += add->temp_blks_read - sub->temp_blks_read;
-         dst->temp_blks_written += add->temp_blks_written - sub->temp_blks_written;
-         INSTR_TIME_ACCUM_DIFF(dst->shared_blk_read_time,
```

1. **Introduce an InstrumentUsage stack** that gets pushed/popped in **InstrStartNode/InstrStopNode**

2. **Add an HLL structure to the stack,** count distinct buffers when EXPLAIN (ANALYZE, BUFFERS DISTINCT) is on

This works:

```
EXPLAIN (ANALYZE, BUFFERS DISTINCT)
SELECT * FROM a JOIN b ON (a.id = b.a_id)
WHERE b.id BETWEEN 500 AND 5000;

Nested Loop  (...)
Buffers: shared hit=13540 hit distinct=0
->  Index Scan using b_id_idx on b  (...)
      Index Cond: ((id >= 500) AND (id <= 5000))
      Index Searches: 1
      Buffers: shared hit=36 hit distinct=36
->  Index Only Scan using a_id_idx on a  (....)
      Index Cond: (id = b.a_id)
      Heap Fetches: 0
      Index Searches: 4501
      Buffers: shared hit=13504 hit distinct=16
```

Estimate: 36 buffers
Actual: 36 buffers

Estimate: 16 buffers
Actual: 13 buffers