# Bitmap Index Only Scan

PgCon 2014

Disclaimer: everything “as far as Markus Winand” knows —which is not a lot in this respect.

# The Problem: Either Or

- Although it is called index-only scan it might still need to fetch (some) rows from the heap (depending on visibility).
- The optimizer might opt for an Bitmap Scan if it seems like IOS would still trigger a lot of random IO to the heap.
- Bitmap Scans generally don't take advantage of the visibility map.

➡Even if Bitmap Scan is better than a Index Only Scan, it will still fetch some tuples from heap although it already fetched the required data from the index before (and forgot about that in the meanwhile).

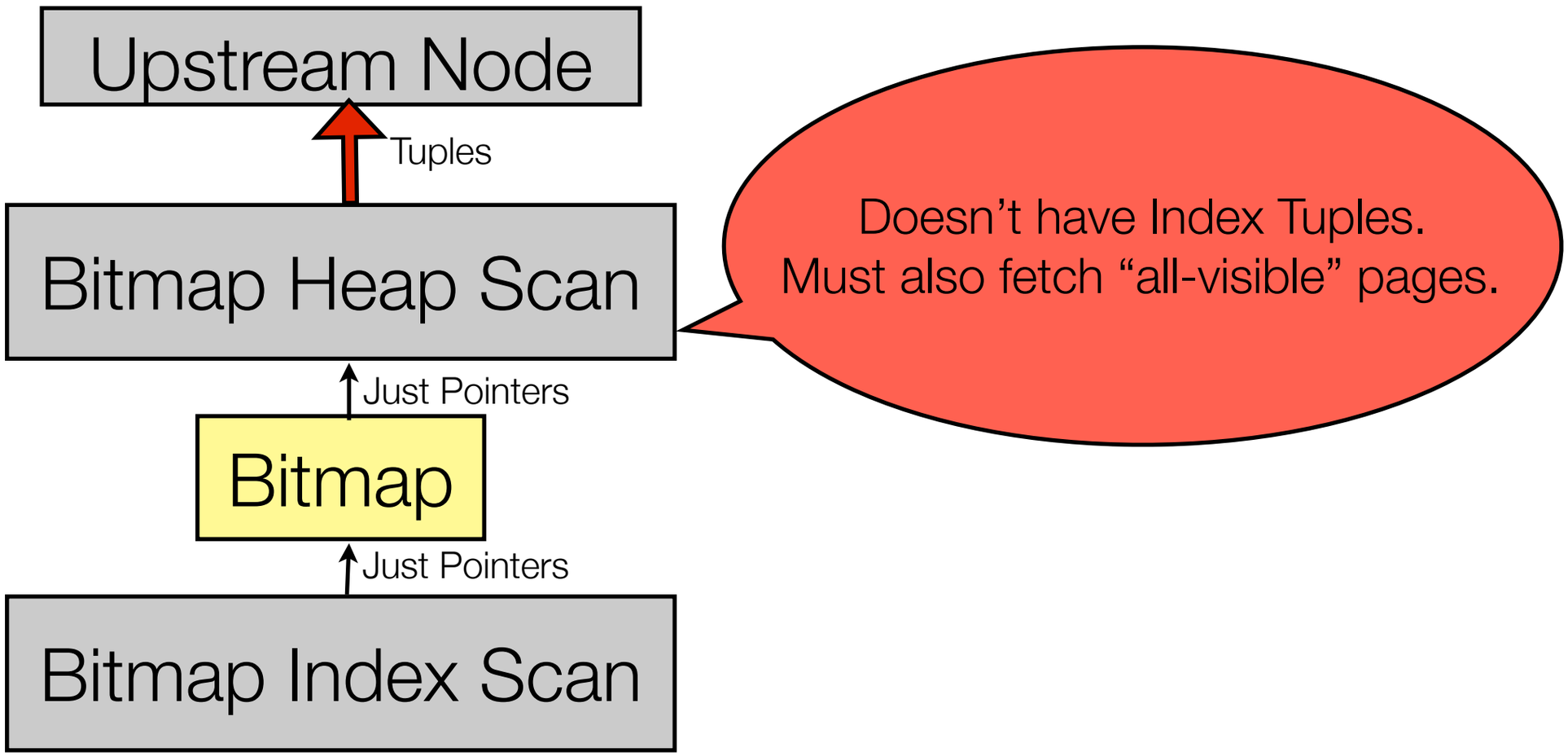
- Right now it is binary:

➡Either avoid reading all-visible pages (IOS) or more efficient IO (Bitmap)

- Does it really need to be that?

# Bitmap Scans: Data Flow

- Bitmap Index Scan puts pointers into bitmap
- Bitmap is passed to Bitmap Heap Scan
- Bitmap Heap Scan gets tuples and sends them upstream.

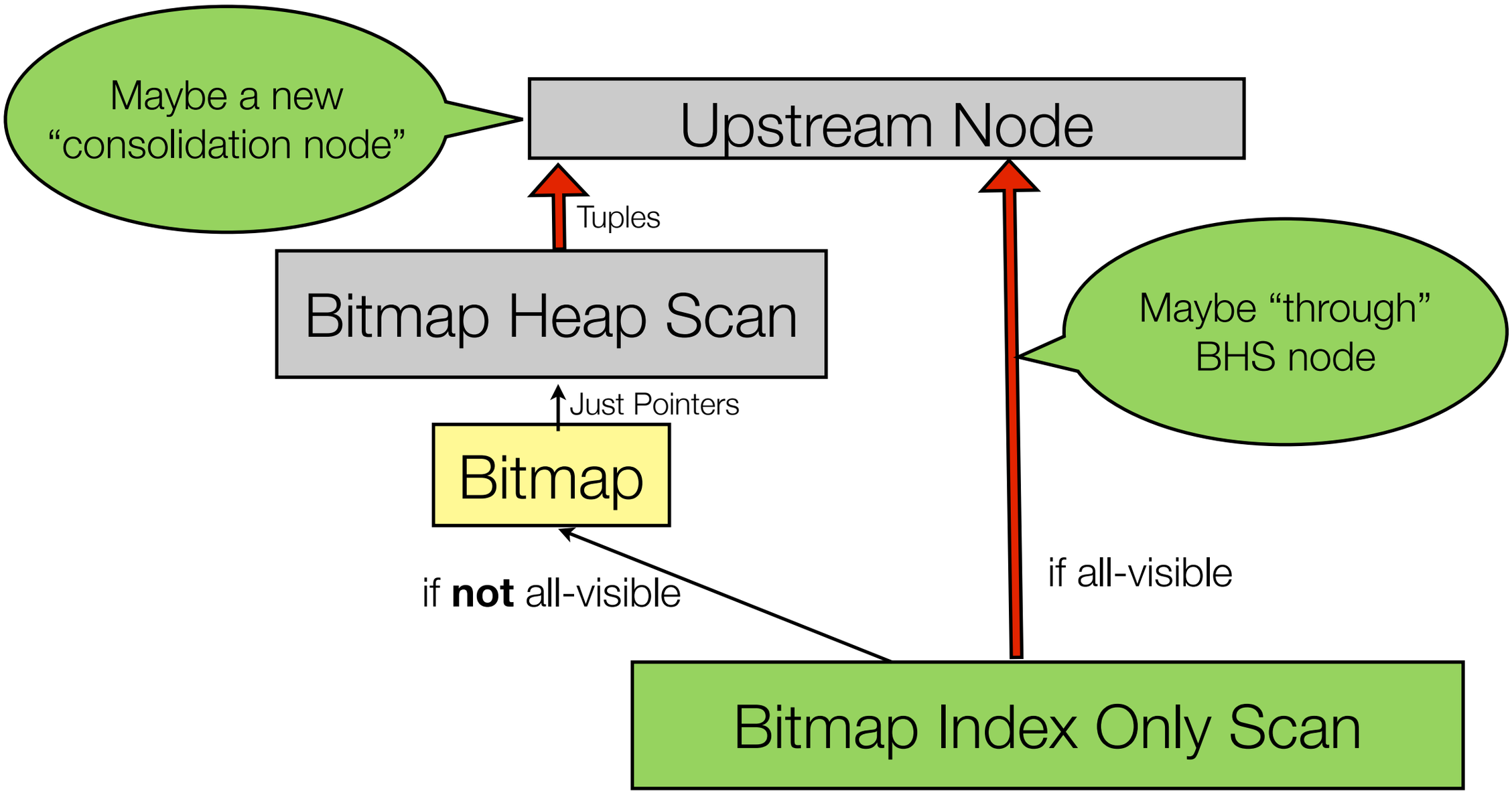


# Idea: Bitmap Index Only Scan

- In case the query would qualify for an IOS, but the planner opts for a Bitmap Scan anyway, why don't we just check the visibility during Bitmap Index Scan and instead of taking note of the heap pointer in the bitmap, just emit the tuple upstream if all-visible?
  - Runtime savings:
    - IO during Bitmap Heap Scan
    - Right-Away emitted tuples don't need space in the Bitmap
    - Rows are emitted earlier (not a strict two-phase execution anymore)
  - Runtime cost:
    - Check the visibility map during Bitmap Index Only Scan
  - Limitations (just wild guess from my side)
    - I guess it makes only sense if there is only one Bitmap Index Scan for the Bitmap Heap Scan — otherwise we would need to store the actual data in the bitmap.

# Bitmap Scans: Proposed Data Flow

- Old path if not all-visible.
- “Shortcut” if all-visible.



# Bitmap Index Only Scan: Challenges

- Challenges
  - What if the lossy bitmap kicks in?
    - Once a row has been emitted directly from the Bitmap Index Scan, it seems to be impossible to switch to the space preserving “lossy” mode (Recheck during Bitmap Heap Scan).
      - Would not be a problem if visibility map isn't cleared during Bitmap Scan (all the tuples in the same heap page would be directly emitted anyways).
        - I suspect Visibility Map can be cleared during Bitmap Scan.
  - I don’t think that it could be made that the Recheck filters the already emitted rows without increasing memory footprint and way more implementation effort.

  ➡That could be a show stopper
-

# Bitmap Index Only Scan: Planning

- Planning
  - Special costing might not be needed if the overhead of checking the visibility map is negligible (I just don't know if it is!)
    - If yes, just do if possible:
      Covering Index AND only one Bitmap Index Scan beneath the Bitmap Heap Scan
  - Even if the visibility map mostly cleared, it might make sense to use Bitmap Scan over Index Scan:
    - If the visibility map is mostly cleared, the Bitmap Index Only Scan would not build up large bitmap (but emit most rows right way, basically behaving like Index Only Scan (other than not being order preserving there is no drawback to the Bitmap Scan plan in that case!)
  - it might be possible to do the planning that without actually increasing the the search space (no more plan permutations — just "a few ifs").