JSONRPC Enhancement in Geth

@jsvisa
AmberGroup

The Project proposal

The Project Proposal(background)

Currently:

- Use debug_traceXXX RPC to get the block traces
- While debug_traceXXX based on the historical statedb to replay the transactions
  - Slow performance
  - High CPU usage

The Project Proposal

Introduce a real-time tracing system, which will

- offer more efficient and prompt insights into the EVM execution
- support indexing the live execution data

How to index the live data

Workflow/Indexing

Support indexing:

- Traces: block number -> traces
- Nonce: (tx sender’s address + nonce) -> Tx Hash

Workflow/Data Schema

kvdb: used to store the newly created block traces and nonces

- A generic Key-Value Database(ethdb)
- Key Schema:
  - Traces: `*BlkNum || BlkHash || TraceType*`
  - Nonce: `Tx Sender's address || nonce`
- Value Schema: RLP encoded data

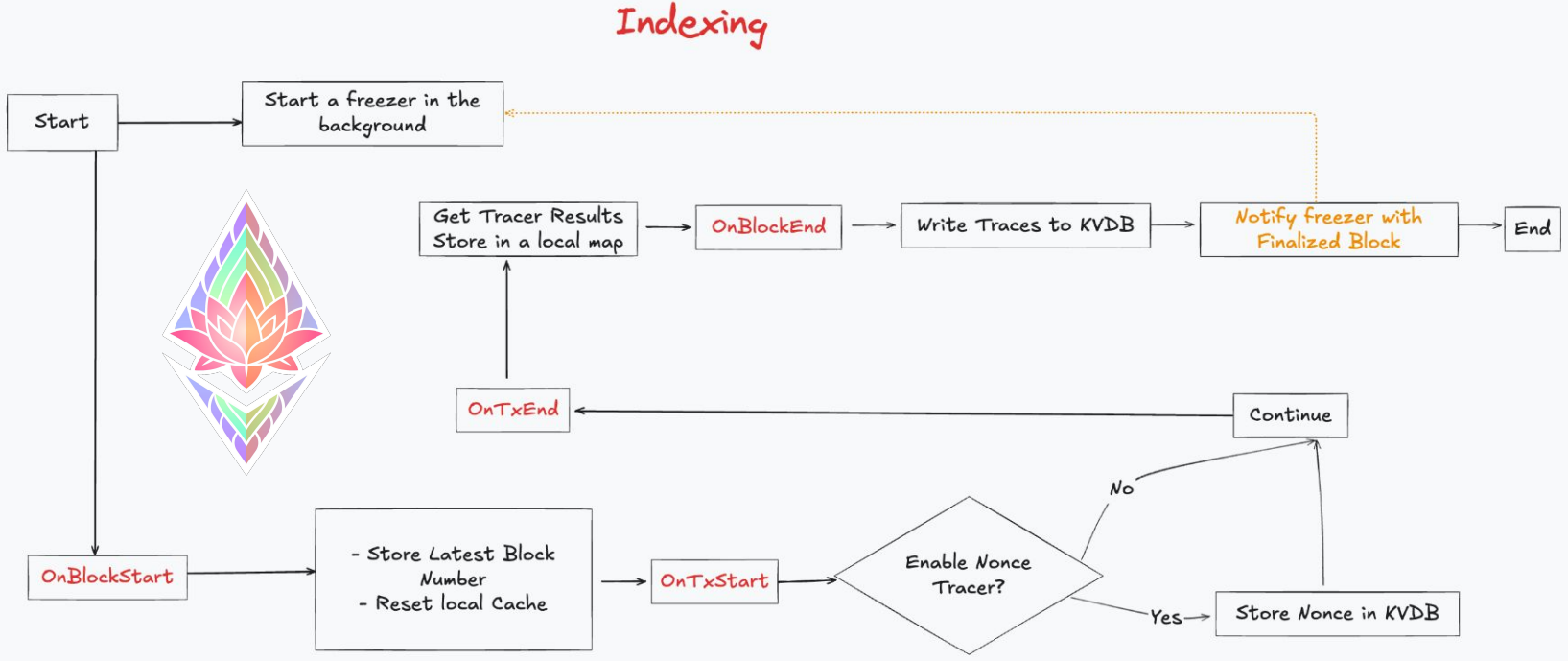
Workflow/Data Schema

kvdb: used to store the newly created block traces and nonces

- A generic Key-Value Database(ethdb)
- Key Schema:
  - Traces: `*BlkNum || BlkHash || TraceType*`
  - Nonce: `Tx Sender's address || nonce`
- Value Schema: RLP encoded data

- TraceType is one of:
  - *callTracer*
  - *flatCallTracer*
  - *parityTracer*
  - *prestateTracer*

Workflow/indexing



How to retrieve the data

Workflow/Retrieving

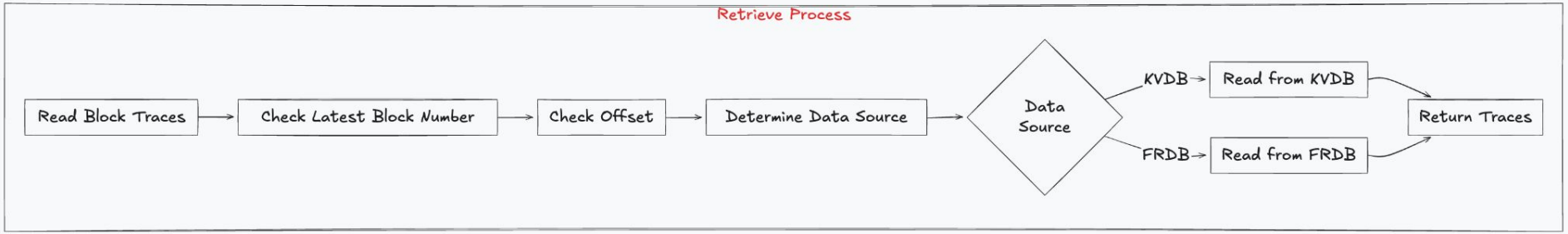
Traces:

- Retrieve by block number/hash
- Retrieve by transaction hash
  - Fetch block number by transaction first -> block number

Nonce:

- Retrieve by sender’s address + nonce

Workflow/Retrieving block traces



How to use this feature

How to use this feature/setup

Haven't merged yet, see #30255 for more detail

```
$ geth ... --http.api=...,trace \
  --vmtrace=live \
  --vmtrace.jsonconfig='{
      "path": "/data/live-trace",
      "enableNonceTracer": true,
      "maxKeepBlocks": 100000,
      "config": {
        "callTracer": { "withLog": true },
        "parityTracer": {},
        "prestateTracer": { "diffMode": true }
      }
}'
```

You need to enable the trace namespace either in http.api or ws.api

Configuration:

- path: a directory used to store the indexed data
- enableNonceTracer: weather or not to indexing the tx sender's tx-hash
- maxKeepBlocks: keep at most N blocks, older ones will be pruned periodly
- config: a map of each tracers you want to trace with, see builtin-tracers for more detail

How to use this feature/retrieve

Supported RPCs:

- trace_block
- trace_filter
- trace_transaction
- eth_getTransactionBySenderAndNonce

How to use this feature/retrieve

Supported RPCs:

- trace_block
- trace_filter
- trace_transaction
- eth_getTransactionBySenderAndNonce

- The trace_* RPCs are compatible with Parity's
- Besides of the Parity's call tracer, we can also retrieve other geth's native tracers' results.

How to use this feature/retrieve

Supported RPCs:

- trace_block
- trace_filter
- trace_transaction
- eth_getTransactionBySenderAndNonce

- The trace_* RPCs are compatible with Parity's
- Besides of the Parity's call tracer, we can also retrieve other geth's native tracers' results.

Eg: get the block(0x1)'s prestateTracer results

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "trace_block",
    "params": [
        "0x1",
        {"tracer": "prestateTracer"}
    ]
}
```

The performance

Space-Time trade-off

The performance between trace_ and debug_trace

Pro:

- No need to re-apply based on a state
- O(1) time complexity to retrieve the data
- Realtime retrieving

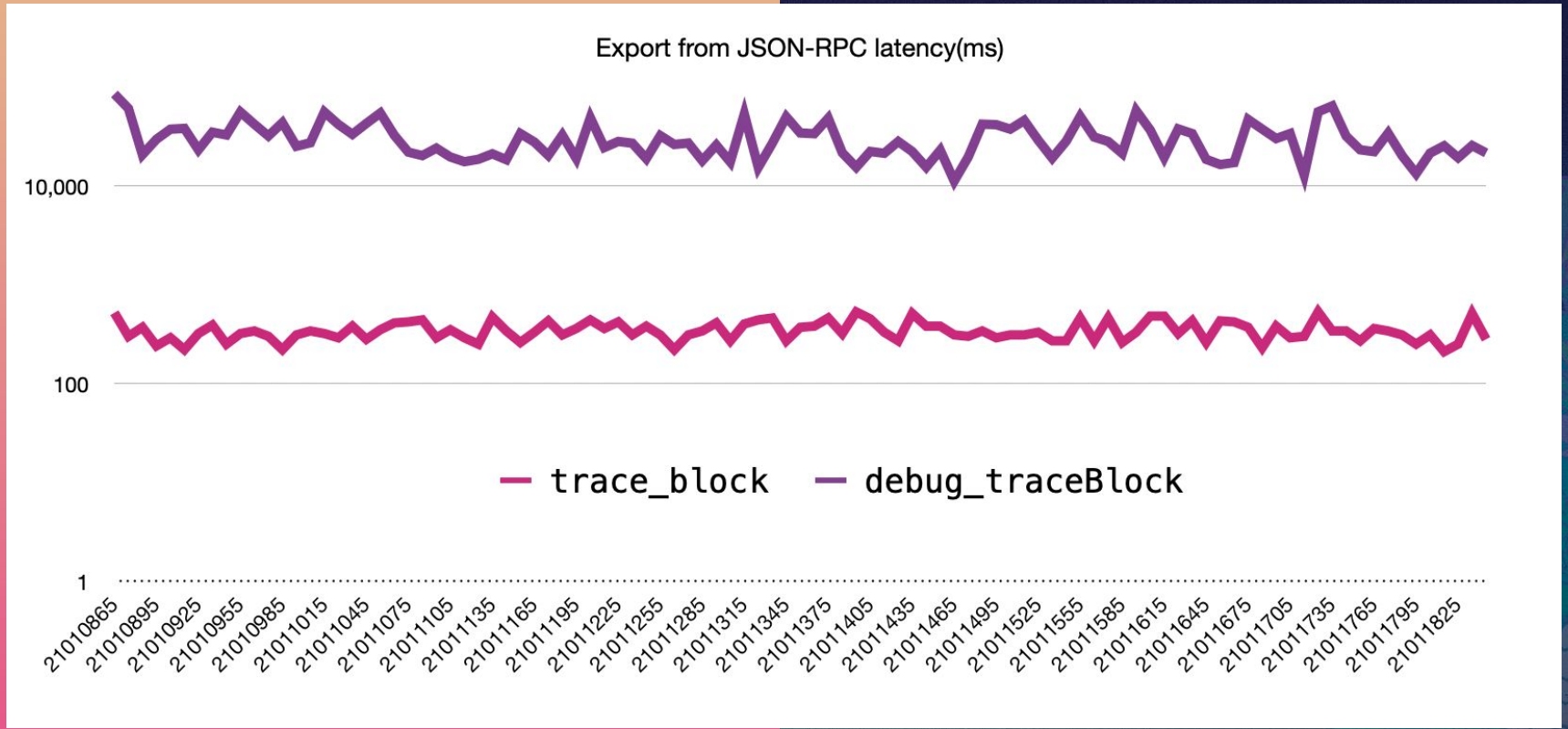
Con:

- Need extra disk space
- Little impact on the on-chain block syncing

Let’s compare the results

Use etl to dump traces from geth RPC into Postgres

```
$ ./etl dump2 \
    --chain=ethereum \
    --lag=64 \
    --start-block=21010865 \
    --end-block=21054092 \
    --period-seconds=60 \
    --max-workers=10 \
    --block-batch-size=10 \
    --batch-size=1 \
    ...
```


Export from JSON-RPC latency(ms)
10,000
100
1
trace_block
debug_traceBlock
21010865
21010895
21010925
21010955
21010985
21011015
21011045
21011075
21011105
21011135
21011165
21011195
21011225
21011255
21011285
21011315
21011345
21011375
21011405
21011435
21011465
21011495
21011525
21011555
21011585
21011615
21011645
21011675
21011705
21011735
21011765
21011795
21011825


Thank you!
@jsvisa
AmberGroup