Mempool Archive

Blocknative actively maintains the most comprehensive historical dataset of mempool transaction events within the Ethereum ecosystem. This collection contains transaction detection events since November 1st, 2019.

- Blocknative logs all mempool transactions from nodes in multiple geographical regions for the Ethereum mainnet blockchain.
- It is updated daily at 13 UTC time with a typical update containing 11M events for a 0.012TB size, though the heaviest days on the network can be as large as 41M events and 0.3TB size.
- This uninterrupted dataset covers major scenarios the network has encountered over the years, including massive surges in traffic, huge gas spikes, bidding wars, the launch of MEV-boost, the price of ETH collapsing, EIP-1559, Black Thursday, and major hacks.
- This data covers 27 data fields, such as gas details, input data, time pending in the mempool, failure reasons, and regional timestamps for each instance seen by our global network of nodes.
- Our self-operated infrastructure provides the earliest detection times from North America, Asia, and Europe.
-

Getting Started

Each date has been partitioned into its own folder named inYYYYMMDD format. Within each date partition, there are 24 files, named by two digit hour (ie 02.csv.gz) that the transaction event was detected in. These files are tab delimited gzipped csvs.

For example, if you wanted to access transactions on June 16th, 2023 from 12pm-1pm, your URL would be: archive.blocknative.com/20230616/12.csv.gz

How to download

Query, download and store the data slices locally using the steps below:

```

Copy curl https://archive.blocknative.com/YYYYMMDD/HH.csv.gz

```

Fetching a full day of data

Here is a script you can use to download all slices in a day on your computer. Just modify with theDATE

```

Copy

# !/bin/bash

# Set the date

DATE="YYYYMMDD" DOMAIN="https://archive.blocknative.com/" BASE_URL="{DOMAIN}{DATE}/"

# Initialize a variable to track successful downloads

SUCCESSFUL_DOWNLOADS=0

# Loop through each hour (00 to 23)

forHOURin{00..23};do

# Construct the URL for the current hour's data

URL="{BASE_URL}{HOUR}.csv.gz"
```

Define the filename for the current hour's data

FILENAME="{HOUR}.csv.gz"

Initialize a variable to keep track of retries

RETRIES=0

Loop to handle retries on 404, 429, and 504 responses

whiletrue;do

Download the data and check the response status code

HTTP_STATUS=(curl-o"FILENAME"-w"%{http_code}" "URL")

Check the status code and print a message

if["HTTP_STATUS"-eq200];then echo"Downloaded FILENAME" ((SUCCESSFUL_DOWNLOADS++)) break# Exit the retry loop on success elif["HTTP_STATUS"-eq429]||["HTTP_STATUS"-eq504];then echo"Received HTTP_STATUS. Retrying in 1 second..." sleep1# Wait for 1 second before retrying ((RETRIES++)) if[RETRIES-ge3];then echo"Retry limit reached. Exiting." exit1 fi elif["HTTP_STATUS"-eq404];then echo"File not found (404). Exiting for FILENAME." break# Exit the retry loop for 404 else echo"Error downloading FILENAME - Status code: HTTP_STATUS" rm"FILENAME"# Remove the empty file break# Exit the retry loop on other errors fi done done

if["SUCCESSFUL_DOWNLOADS"-eq24];then echo"All slices downloaded successfully!" else echo"Some slices were not downloaded successfully." fi

```

Save this script to a file, for example,download_slices.sh , and make it executable using the following command:

```

Copy chmod +x download_slices.sh

```

Then, run the script by executing:

```

Copy ./download_slices.sh

```

Fetching on a custom range

Here is a script you can use to (1) download all hourly slices in a specific range of days on your computer, or (2) all specific hourly slices on a specific day.

Options:

1. -date-range
2. : for downloading full hourly slices for all days within this range (both dates inclusive). Format:YYYYMMDD-YYYYMMDD
3. 
    - For date range:./download_mempool.sh --date-range YYYYMMDD-YYYYMMDD
4. *
5. -hour-range
6. : for downloading data for specific hours on a particular day. Format:YYYYMMDD:HH-HH
7. 
    - For hour range:./download_mempool.sh --hour-range YYYYMMDD:HH-HH
8. *
9. 
```

```

Copy

# !/bin/bash

# Fetch arguments

while[[ #-gt0]];do key="1" casekeyin --date-range) DATE_RANGE="2" shift;shift ;; --hour-range) HOUR_RANGE="2" shift;shift ;; *) shift ;; esac done

DOMAIN="https://archive.blocknative.com/" SUCCESSFUL_DOWNLOADS=0

download_data(){ localDATE=1 localHOUR_START=2 localHOUR_END=3 localBASE_URL="{DOMAIN}{DATE}/"

forHOURin(seq-wHOUR_START HOUR_END);do URL="{BASE_URL}{HOUR}.csv.gz" FILENAME="{DATE}_{HOUR}.csv.gz" RETRIES=0

whiletrue;do HTTP_STATUS=(curl-o"FILENAME"-w"%{http_code}" "URL")

if["HTTP_STATUS"-eq200];then echo"Downloaded FILENAME" ((SUCCESSFUL_DOWNLOADS++)) break elif["HTTP_STATUS"-eq429]||["HTTP_STATUS"-eq504];then echo"Received HTTP_STATUS. Retrying in 1 second..." sleep1 ((RETRIES++)) if[ RETRIES-ge3];then echo"Retry limit reached. Exiting." exit1 fi elif["HTTP_STATUS"-eq404];then echo"File not found (404). Exiting for FILENAME." break else echo"Error downloading FILENAME - Status code: HTTP_STATUS" rm"FILENAME" break fi done done }

# Date Range Mode

if[!-z"DATE_RANGE"];then IFS='-'read-raDATES<<<"DATE_RANGE" START_DATE={DATES[0]} END_DATE={DATES[1]}

forDATEin(seq-wSTART_DATE END_DATE);do download_dataDATE0023 done fi

# Hour Range Mode

if[!-z"HOUR_RANGE"];then IFS=':'read-raPARTS<<<"HOUR_RANGE" DATE={PARTS[0]} IFS='-'read-raHOURS<<<"{PARTS[1]}" HOUR_START={HOURS[0]} HOUR_END={HOURS[1]}

download_dataDATE HOUR_START HOUR_END fi

if["SUCCESSFUL_DOWNLOADS"-gt0];then echo"All slices downloaded successfully!" else echo"Some slices were not downloaded successfully." fi

```

Save this script to a file, for example,download_mempool.sh , and make it executable using the following command:

```

Copy chmod +x download_mempool.sh

```

Then, run the script by executing the command specified above the script.

Data Schema

Blocknative logs all mempool transactions from nodes in multiple geographical regions for the Ethereum mainnet blockchain. The Archive contains historic events for all transactions:

- entering the mempool
- denied entry into the mempool (rejection with reason)
- exiting the mempool (eviction with reason)
- replacing existing mempool transaction (speedup or cancel)
- finalized on chain (confirmed or failed)
-

The number of times a transaction appears in the Archive corresponds to the number of status changes it undergoes. The

detecttime

field indicates the time when the status change was first observed. Below you can find the complete schema for the data:

Field Name Description Data Type Example detecttime Timestamp that the transaction was detected in mempool. timestamp 2020-03-12 00:00:00.409000 hash Unique identifier hash for a given transaction. string 0x6b4104838fd153b2d1ab705737843f5ea99666794391dd52653960970dc7e5ef status Status of the transaction. string Pending ,speedup ,cancel ,failed ,stuck ,dropped ,confirmed ,evicted ,rejected region The geographic region for the node that detected the transaction. string us-east-1 ,eu-central-1 ,ap-southeast-1 reorg If there was a reorg, refers to the blockhash of the reorg. string 0xf2ec4b2a7b951e4400e99d1171c4fb875fd388b15b6cb97bf5ad1c8dbea3a73a replace If the transaction was replaced (speedup/cancel), the transaction hash of the replacement. string 0xcea6244a7f0a7c2630085ca3e47e1ecfc28a5c03a08a8f3ec5f43fbef3d83dd5 curblocknumber The block number the event was detected in. decimal(18,0) 12429202 failurereason If a transaction failed, this field provides contextual information. string Reverted: ""UniswapV2Router: INSUFFICIENT_OUTPUT_AMOUNT"" blockspending If a transaction was finalized (confirmed, failed), this refers to the number of blocks that the transaction was waiting to get on-chain. int 2 timepending If a transaction was finalized (confirmed, failed), this refers to the time in milliseconds that the transaction was waiting to get on-chain. bigint 4678 nonce A unique number which counts the number of transactions sent from a given address. decimal(38,0) 27744 gas The maximum number of gas units allowed for the transaction. decimal(38,0) 55588 gasprice The price offered to the miner/validator per unit of gas. Denominated in wei. decimal(38,0) 1200000000 value The amount of ETH transferred or sent to contract. Denominated in wei. decimal(38,0) 147940000000000 toaddress The destination of a given transaction. string 0x501c885e8f519feeb1a8f9429ea586ebd378b549 fromaddress The source/initiator of a given transaction. string 0xf974334a62b3aab3e2b5509f65b9b2141d8efa03 input Additional data that can be attached to a transaction. This field can be used to tell a smart contract to execute a function. string 0xa9059cbb000000000000000000000000955a0ef4e120528f8486c04c97388d530cfbf239000000000000000000 network The specific Ethereum network used. string bsc-main ,goerli ,kovan ,main ,rinkeby ,ropsten ,xdai type Post EIP-1559, this indicates how the gas parameters are submitted to the network: - type 0 - legacy

- type 1 - usage of access lists according to EIP-2930
- type 2 - usingmaxpriorityfeepergas andmaxfeepergas int 0 ,1 ,2 maxpriorityfeepergas The maximum value for a tip offered to the miner/validator per unit of gas. The actual tip paid can be lower if (maxfee -basefee) <maxpriorityfee . Denominated in wei. decimal(38,0) 111373960022 maxfeepergas The maximum value for the transaction fee (includingbasefee and tip) offered to the miner/validator per unit of gas. Denominated in wei. decimal(38,0) 111373960022 basefeepergas The fee per unit of gas paid and burned for thecurblocknumber . This fee is algorithmically determined. Denominated in wei. decimal(38,0) 111373960022 dropreason If the transaction was dropped from the mempool, this describes the contextual reason for the drop. string unexecutable-txs ,unpayable-txs ,replaced-txs ,account-cap-txs ,old-txs ,underpriced-txs ,low-nonce rejectionreason If the transaction was rejected from the mempool, this describes the contextual reason for the rejection. string exceeds block gas limit ,insufficient funds for gas * price + value intrinsic gas too low ,non transaction ,underpriced stuck A transaction was detected in the queued area of the mempool and is not eligible for inclusion in a block. boolean 1 gasused If the transaction was published on-chain, this value indicates the amount of gas that was actually consumed. Denominated in wei. decimal(38,0) 111373960022 detect_date A truncated version ofdetecttime . Best used as a partition for large datasets and as a search parameter to speed up queries. string 2023-10-10 blobversionedhashes String representation of versioned blob hashes associated with the transaction's EIP-4844 data blobs. string 0x01f3ee17d9bd3b1e37df90813b95b21ec3504d66c5fe52974712bc4efb7db300 maxfeeperblobgas The maximum total fee per blob gas the sender is willing to pay for blob gas in wei decimal(38,0) 242082408240

Frequently Asked Questions

What attribution must I provide when using the Blocknative Data Archive?

The archive is publicly available according to open data standards and licenses datasets under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International.

2.1 Attribution — End Users must give appropriate credit, provide a link to the license, and indicate if changes were made. End Users may do so in any reasonable manner, but not in any way that suggests the licensor endorses End Users or their use.2.2 NonCommercial — End Users may not use the material for commercial purposes.2.3 ShareAlike — If End Users remix, transform, or build upon the material, End Users must distribute their contributions under the same license as the original.

Please use the following as a guideline for attribution:

1. Papers
2. : Data provided byBlocknative
3. ?
4. ?
5.

If you have any questions please reach out to us orDiscord .

What format is the data?

The data is stored in hourly slices with file format*.csv.gz The data is tab delimited.

How many nodes are gathering mempool data?

We run highly redundant node infrastructure in each region to ensure strong uptime.

How can I identify on-chain transactions?

On-chain transaction have aconfirmed status.

```

Copy SELECT* FROMmempool_archive WHEREstatus='confirmed'

```

How can I identify private transactions?

A private transaction does not have apending event.timepending is determined from the difference between a transaction'spending event andconfirmed event.

```

Copy SELECT* FROMmempool_archive WHEREtimepending=0 ANDstatus='confirmed'

```

What is the difference between dropreason and rejectionreason ?

A dropped transaction might have been valid but deemed less important or lower-priority. A rejected transaction is one that is fundamentally flawed or invalid according to the Ethereum protocol rules.

A drop reason could be that there isn't enough ETH in the EOA to cover gas fees. A rejection reason could be incorrect transaction signatures.

Dropped transactions existed in the mempool, but are dropped to make room for incoming transactions. Rejected transactions never make it to the mempool.

Previous Blocknative Data Archive Next Blob Archive Last updated4 days ago

Was this helpful?