Pruning

Overview

Pruning pertains to eliminating or cleaning obsolete historical data to optimize disk space. Clients are responsible for maintaining the world state, which comprises a database that portrays the current Ethereum network status. The world state encompasses accounts, contracts, and other information.

Pruning aims to reduce disk requirements by storing only the current world state and removing historical data. This differs from archive nodes that retain complete transaction and state history. Pruning is helpful for users who don't require historical data and prefer to interact solely with the current state of the network. However, pruning may limit the client's ability to fulfill requests that depend on historical information. Nethermind provides two kinds of pruning –full pruning andin-memory pruning ; both are enabled by default, also calledhybrid pruning .

How it works

During synchronization using the snap sync method, Nethermind produces a local copy of the Ethereum network state, which typically amounts to approximately 160 GB (state database). The entire database may rise to 600–650GB. Although this size increases by around 30 GB each week, specific historical data is retained that is not necessary for node operation or to maintain the current Ethereum state.

When full pruning is activated and initiated, a thorough examination of the entire state tree is conducted to determine which data is no longer required and can be treated as historical. It then determines which information corresponds to the current state and duplicates it alongside the existing version. During verification of each node in the state, the new pruned state replaces the previous one. Once the verifier confirms everything is functioning correctly, the old state database is eliminated, resulting in significant savings in disk space. As a result, the size of the database will be close to its initial size again.

In-memory pruning is a continuous process that occurs under regular operation. Instead of saving a new state on each block, Nethermind will keep it in memory until a certain threshold is reached. At that point, Nethermind will only store data required by the newer state and discard unnecessary ones. This significantly reduces the total amount of data written while improving block processing performance. In-memory pruning is independent of full pruning.

Preparation for full pruning

Because full pruning is executed while the node is connected to the network, it can affect the node's performance. The process can consume significant memory, CPU, and disk resources, impacting block processing time. This, in turn, can lead to reduced rewards for validator setups, particularly for proof-of-stake chains.

Currently, full pruning takes between 20 to 30 hours to complete, although the duration may vary based on the hardware configuration. At least 200 GB of storage is required to ensure pruning runs smoothly. Nevertheless, it's recommended to have 300 GB or more.

Important Do not turn on full pruning on an archive node, as these are two opposing features. Archive nodes are designed to store complete historical data, whereas full pruning eliminates it. Use the option--Pruning.Mode None to ensure that pruning is turned off completely.

Configuring full pruning

As a very first point, check out thepruning configuration options . To activate full pruning, use either the--Pruning.Mode Hybrid or--Pruning.Mode Full command line options.

info Setting--Pruning.Mode Hybrid enables bothInMemory andFull modes. TheInMemory mode helps the node storage grow slower than theFull mode. Because of this, full pruning is executed less frequently, promoting healthier disk operation. Since full pruning is hardware-intensive, this configuration also benefits attestation results. The next step is to determine the trigger conditions for full pruning. Currently, there are 3 options available:

- Manual
- State database size threshold
- Remaining storage space threshold

Manual

Manual mode triggers full pruning only upon request, providing complete control. To configure this mode, use the following options:--Pruning.Mode Hybrid --Pruning.FullPruningTrigger Manual .

Pruning can also be triggered using theadmin_prune JSON-RPC method. Here's how to configure it:

- Add theadmin
- namespace to--JsonRpc.EnabledModules
- . For instance,
- --JsonRpc.EnabledModules [eth,net,...,admin]
- Create a separate port foradmin
- namespace only:
- --JsonRpc.AdditionalRpcUrls http://localhost:8555|http|admin

Restart the client, and if everything is configured correctly, you should be able to call theadmin_prune method, and full pruning should start.

info One potential disadvantage of the manual mode is that if full pruning is not triggered on time, it cannot be performed later because of insufficient disk space. In such cases, the only option to free up the disk space is to resync the node from scratch.

State database size threshold

This mode triggers full pruning automatically when the state database reaches the specified size. To enable this mode, use the following options:--Pruning.Mode Hybrid --Pruning.FullPruningTrigger StateDbSize --Pruning.FullPruningThresholdMb 256000 , where the value ofPruning.FullPruningThresholdMb should be set based on your requirements.

The above configuration triggers full pruning automatically whenever the state database size exceeds 256,000 MB (250 GB). Assuming the state database has initially around 160 GB, pruning will be triggered when the database size grows by more than 90 GB.

tip To avoid unexpected behavior and ensure that full pruning can be completed in full, it's recommended to set the threshold value to trigger pruning before the free disk space drops below 250 GB as a minimum. This ensures that sufficient free disk space is available for the pruning process.

Remaining storage space threshold

note This is the recommended approach as it ensures that pruning is executed on time. This mode triggers full pruning when the storage space reaches the specified minimum. To enable this mode, use the following options:--Pruning.Mode Hybrid --Pruning.FullPruningTrigger VolumeFreeSpace --Pruning.FullPruningThresholdMb 256000 , where the value ofPruning.FullPruningThresholdMb should be set based on your requirements. However, it should not be set below the default value of256000 .

The above configuration triggers full pruning whenever free disk space drops to 256,000 MB (250 GB) or below. This ensures that pruning is invoked as infrequently as possible while also ensuring that sufficient free storage is always available to trigger it.

warning It's recommended not to set the value below 250 GB for stability reasons. In reality, full pruning should require approximately the same amount of storage as a fresh state database (around 160 GB). Still, it also needs a threshold for processing and other operations since it creates a copy of the existing state database. Therefore, it's essential to maintain a minimum amount of storage to ensure a robust pruning performance.

Monitoring progress

When full pruning is triggered correctly, the corresponding messages appear in the Nethermind logs.

The very first ones should be:

Full Pruning Ready to start: pruning garbage before statewith root . WARN: Full Pruning Started on root hash : do not close the node until finished or progress will be lost. From that moment, ensure that no restarts will be performed on Nethermind to ensure that full pruning runs correctly. After a few minutes first logs with progress would start to appear. For instance:

Full Pruning In Progress: 00:00:57.0603307 1.00 mln nodes mirrored. Full Pruning In Progress: 00:01:40.3677103 2.00 mln nodes mirrored. Full Pruning In Progress: 00:02:25.6437030 3.00 mln nodes mirrored. Pruning may take evenmore than 30 hours , depending on hardware configuration.

When pruning is completed, there is a log message similar the following:

Full Pruning Finished: 15:25:59.1620756 1,560.29 mln nodes mirrored. As you may notice, it took around 15 hours in this example.

info Since the amount of mirrored nodes is not a static value, providing a simple progress indicator in percentage is impossible. Therefore, the approximate value given should be used to determine when full pruning is expected to be completed.

Additional settings

Memory budget

ThePruning.FullPruningMemoryBudgetMb configuration option controls the memory budget allocated for the trie visit during the full pruning process. During pruning, pending nodes are queued to a pool of nodes whose size is determined by this value. This allows multiple nodes to share a single I/O. By increasing this value, the required read IOP per second can be significantly reduced, resulting in a faster full pruning operation. However, this improvement comes at the expense of increased memory usage.

Assuming your system has 64GB of RAM, with Nethermind, the consensus client, and system expenses consuming 20GB, you need to determine the maximum value forPruning.FullPruningMemoryBudgetMb while ensuring the system remains stable and respects the given limit. In this case, you have 44 GB (64 GB - 20 GB) of available memory for increasing thePruning.FullPruningMemoryBudgetMb value. To calculate the maximum value in MB, multiply the available memory by 1024:44 GB * 1024 = 45,056 MB

info Because of the workload on Ethereum Mainnet, setting the value ofPruning.FullPruningMemoryBudgetMb higher than 16 GB may not provide any additional performance benefits. Please note that this example assumes the remaining 44 GB of memory is sufficient for the full pruning memory budget. Depending on the specific use case and system requirements, it may be necessary to adjust the value further to optimize performance and resource usage.

Pruning completion behavior

ThePruning.FullPruningCompletionBehavior configuration option determines Nethermind's behavior after full pruning is completed. By default, Nethermind will continue to progress as usual. However, if a user wishes to shut down the node after pruning, there are three options available:

- None
- : No action taken
- ShutdownOnSuccess
- : Nethermind shuts down if pruning succeeds
- AlwaysShutdown
- : Nethermind shuts down once pruning completes, regardless of whether it succeeds or fails

Number of pruning concurrent tasks

ThePruning.FullPruningMaxDegreeOfParallelism configuration option determines the number of parallel tasks/threads that can be used by pruning:

- -1
- : uses the number of logical processors
- 0
- : uses 25% of logical processors
- 1
- : runs on a single thread

The recommended value depends on the type of node being used. If the node needs to be responsive (e.g., RPC or validator), using a value below the number of logical processors is recommended. The default value is recommended if the node doesn't have many other responsibilities but needs to follow the chain without delays and produce live logs reliably. If the node doesn't need to be responsive, has very fast I/O (such as NVMe), and the shortest pruning time is desired, this can be set to 2–3 times the number of logical processors.

In-memory cache size

ThePruning.CacheMb configuration option determines the size, in MB, of the memory pool of nodes used for in-memory pruning. The default value is 1024. Increasing this value can help reduce the rate at which the state database grows.

Side notes

For pruning, keep in mind the following:

- Full pruning is a cumbersome task, but it's performed in the background, so the node continues progressing and following the chain.
- The process' heaviness may affect the effectiveness of the validator rewards. Still, since it's executed only once every few months, it shouldn't have a significant impact on overall results (we've xperienced approximately 5–10% loss of rewards during full pruning).
- Ensure that your storage has at least 250 GB of free space after syncing the node. Otherwise, full pruning will never complete successfully.

- Several things can be done to reduce the size of the database after syncing: settingSync.AncientBodiesBarrier
- andSync.AncientReceiptsBarrier
- to a proper value higher than 0, using a consensus client that requires less storage, and setting logs to the lowest level to avoid log spamming. Edit this page Last updatedonMar 26, 2024 Previous Database Next Performance tuning