aio-libs / **aiohttp** Public

Code | Issues 184 | Pull requests 90 | Discussions | Actions | Security and quality 33

Commit c4d77c3

Dreamsorcerer and **gonas** authored on Feb 22 · 35 / 39 · Verified

```
Bound DNS cache (#12106) (#12117)

(cherry picked from commit 8ab84c5)

---------

Co-authored-by: gonas <nhjangh@gamil.com>
```

master (#12117) · v3.13.5 v3.13.4

1 parent 7d590fe commit c4d77c3

3 files changed +95 -6 lines changed

Top

Filter files...

- CHANGES
  - 12106.feature.rst
- aiohttp
  - connector.py
- tests
  - test_connector.py

3 files changed +95 -6 lines changed

Search within code

CHANGES/12106.feature.rst

```
@@ -0,0 +1 @@
+ Added a ``dns_cache_max_size`` parameter to ``TCPConnector`` to limit the size of the cache --
  by :user:`Dreamsorcerer`.
```

aiohttp/connector.py

```
@@ -856,25 +856,33 @@ async def _create_connection(
856 856
857 857
858 858     class _DNSCacheTable:
859     -     def __init__(self, ttl: Optional[float] = None) -> None:
```

```
860     -           self._addrs_rr: Dict[Tuple[str, int], Tuple[Iterator[ResolveResult], int]] = {}
    859 +       def __init__(self, ttl: Optional[float] = None, max_size: int = 1000) -> None:
    860 +           self._addrs_rr: OrderedDict[
    861 +               Tuple[str, int], Tuple[Iterator[ResolveResult], int]
    862 +           ] = OrderedDict()
861 863             self._timestamps: Dict[Tuple[str, int], float] = {}
862 864             self._ttl = ttl
    865 +           self._max_size = max_size
863 866
864 867         def __contains__(self, host: object) -> bool:
865 868             return host in self._addrs_rr
866 869
867 870         def add(self, key: Tuple[str, int], addrs: List[ResolveResult]) -> None:
    871 +           if key in self._addrs_rr:
    872 +               self._addrs_rr.move_to_end(key)
    873 +
868 874             self._addrs_rr[key] = (cycle(addrs), len(addrs))
869 875
870 876             if self._ttl is not None:
871 877                 self._timestamps[key] = monotonic()
872 878
    879 +           if len(self._addrs_rr) > self._max_size:
    880 +               oldest_key, _ = self._addrs_rr.popitem(last=False)
    881 +               self._timestamps.pop(oldest_key, None)
    882 +
873 883         def remove(self, key: Tuple[str, int]) -> None:
874 884             self._addrs_rr.pop(key, None)
875     -
876     -           if self._ttl is not None:
877     -               self._timestamps.pop(key, None)
    885 +           self._timestamps.pop(key, None)
878 886
879 887         def clear(self) -> None:
880 888             self._addrs_rr.clear()
        @@ -885,6 +893,7 @@ def next_addrs(self, key: Tuple[str, int]) -> List[ResolveResult]:
885 893             addrs = list(islice(loop, length))
886 894             # Consume one more element to shift internal state of `cycle`
887 895             next(loop)
    896 +           self._addrs_rr.move_to_end(key)
888 897             return addrs
889 898
890 899         def expired(self, key: Tuple[str, int]) -> bool:
        @@ -973,6 +982,7 @@ def __init__(
973 982             fingerprint: Optional[bytes] = None,
974 983             use_dns_cache: bool = True,
975 984             ttl_dns_cache: Optional[int] = 10,
```

```
                985 +         dns_cache_max_size: int = 1000,
 976            986           family: socket.AddressFamily = socket.AddressFamily.AF_UNSPEC,
 977            987           ssl_context: Optional[SSLContext] = None,
 978            988           ssl: Union[bool, Fingerprint, SSLContext] = True,
                      @@ -1011,7 +1021,9 @@ def __init__(
1011           1021               self._resolver_owner = False
1012           1022
1013           1023           self._use_dns_cache = use_dns_cache
1014                 -         self._cached_hosts = _DNSCacheTable(ttl=ttl_dns_cache)
                1024 +         self._cached_hosts = _DNSCacheTable(
                1025 +             ttl=ttl_dns_cache, max_size=dns_cache_max_size
                1026 +         )
1015           1027           self._throttle_dns_futures: Dict[
1016           1028               Tuple[str, int], Set["asyncio.Future[None]"]
1017           1029           ] = {}
```

tests/test_connector.py

```
                      @@ -4036,6 +4036,25 @@ async def handler(request):
4036           4036
4037           4037
4038           4038   class TestDNSCacheTable:
                4039 +     host1 = ("localhost", 80)
                4040 +     host2 = ("foo", 80)
                4041 +     result1: ResolveResult = {
                4042 +         "hostname": "localhost",
                4043 +         "host": "127.0.0.1",
                4044 +         "port": 80,
                4045 +         "family": socket.AF_INET,
                4046 +         "proto": 0,
                4047 +         "flags": socket.AI_NUMERICHOST,
                4048 +     }
                4049 +     result2: ResolveResult = {
                4050 +         "hostname": "foo",
                4051 +         "host": "127.0.0.2",
                4052 +         "port": 80,
                4053 +         "family": socket.AF_INET,
                4054 +         "proto": 0,
                4055 +         "flags": socket.AI_NUMERICHOST,
                4056 +     }
                4057 +
4039           4058       @pytest.fixture
4040           4059       def dns_cache_table(self):
4041           4060           return _DNSCacheTable()
                      @@ -4121,6 +4140,63 @@ def test_next_addrs_single(self, dns_cache_table) -> None:
```

```
4121  4140            addrs = dns_cache_table.next_addrs("foo")
4122  4141            assert addrs == ["127.0.0.1"]
4123  4142
      4143  +     def test_max_size_eviction(self) -> None:
      4144  +         table = _DNSCacheTable(max_size=2)
      4145  +
      4146  +         table.add(self.host1, [self.result1])
      4147  +         table.add(self.host2, [self.result2])
      4148  +
      4149  +         host3 = ("example.com", 80)
      4150  +         result3: ResolveResult = {
      4151  +             **self.result1,
      4152  +             "hostname": "example.com",
      4153  +             "host": "1.2.3.4",
      4154  +         }
      4155  +         table.add(host3, [result3])
      4156  +
      4157  +         assert len(table._addrs_rr) == 2
      4158  +         assert self.host1 not in table._addrs_rr
      4159  +         assert host3 in table._addrs_rr
      4160  +
      4161  +     def test_lru_eviction(self) -> None:
      4162  +         table = _DNSCacheTable(max_size=2)
      4163  +
      4164  +         table.add(self.host1, [self.result1])
      4165  +         table.add(self.host2, [self.result2])
      4166  +
      4167  +         table.next_addrs(self.host1)
      4168  +
      4169  +         host3 = ("example.com", 80)
      4170  +         result3: ResolveResult = {
      4171  +             **self.result1,
      4172  +             "hostname": "example.com",
      4173  +             "host": "1.2.3.4",
      4174  +         }
      4175  +         table.add(host3, [result3])
      4176  +
      4177  +         assert self.host1 in table._addrs_rr
      4178  +         assert self.host2 not in table._addrs_rr
      4179  +
      4180  +     def test_lru_eviction_add(self) -> None:
      4181  +         table = _DNSCacheTable(max_size=2)
      4182  +
      4183  +         table.add(self.host1, [self.result1])
      4184  +         table.add(self.host2, [self.result2])
      4185  +
```

```
4186 +         # Re-add, thus making host1 the most recently used.
4187 +         table.add(self.host1, [self.result1])
4188 +
4189 +         host3 = ("example.com", 80)
4190 +         result3: ResolveResult = {
4191 +             **self.result1,
4192 +             "hostname": "example.com",
4193 +             "host": "1.2.3.4",
4194 +         }
4195 +         table.add(host3, [result3])
4196 +
4197 +         assert self.host1 in table._addrs_rr
4198 +         assert self.host2 not in table._addrs_rr
4199 +
4124 4200
4125 4201     async def test_connector_cache_trace_race():
4126 4202         class DummyTracer:
```

Comments 0

Please sign in to comment.