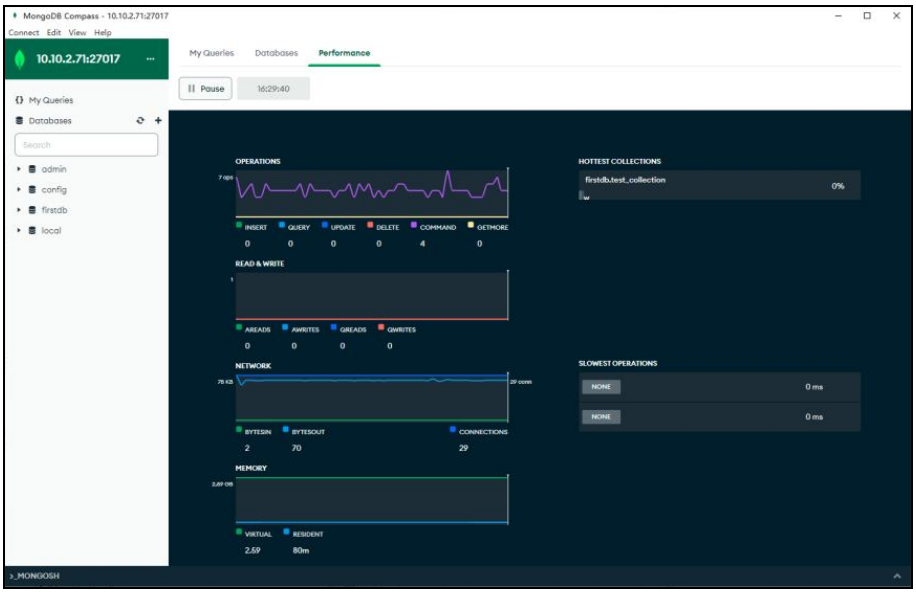

图 7-4-1　查看性能

MongoDB Compass 监控指标说明，见表 7-4-2。

表 7-4-2　MongoDB Compass 监控指标说明

图表/表格	描述
Operations	显示数据库处理的各类操作数量，与 mongostat 相关
Read & Write	显示活动读取数、排队读取数、活动数写入、排队写入
Network	显示数据库操作的网络性能
Memory	显示内存统计信息
Hottest Collections	显示报告的活动最多的集合。集合对应于图表中的给定时刻；即当在图表上移动时，相应的集合在表格中突出显示/显示
Slowest Operations	显示最慢的操作。集合对应于图表中的给定时刻；即当在图表上移动时，相应的集合在表格中突出显示/显示

任务五　使用 Percona Monitoring and Management 监控 MongoDB

【任务介绍】

扫码看视频

Percona Monitoring and Management（简称“PMM”）是一款开源数据库监控和管理工具软件，支持 MongoDB、MySQL、MariaDB、PostgreSQL 等常用数据库

管理系统。PMM 内置监控指标和性能分析模型，并具有可视化监控分析功能。

本任务通过部署 PMM，实现 MongoDB 数据库运行状况的监控与故障排除，提高数据库的性能。

【任务目标】

（1）实现 PMM 的搭建。

（2）实现 MongoDB 的监控与管理。

【操作步骤】

步骤 1：准备 openEuler 服务器。

（1）创建虚拟机并完成 openEuler 的安装。在 VirtualBox 中创建虚拟机，完成 openEuler 的安装。虚拟机与操作系统的配置信息见表 7-5-1，注意虚拟机网卡的工作模式为桥接。

表 7-5-1　虚拟机与操作系统配置

虚拟机配置	操作系统配置
虚拟机名称：VM-Project-07-Task-05-10.10.2.75	主机名：Project-07-Task-05
内存：1GB	IP 地址：10.10.2.75
CPU：1 颗 1 核心	子网掩码：255.255.255.0
虚拟硬盘：20GB	网关：10.10.2.1
网卡：1 块，桥接	DNS：8.8.8.8

（2）完成虚拟机的主机配置、网络配置及通信测试。启动并登录虚拟机，依据表 7-5-1 完成主机名和网络的配置，能够访问互联网和本地主机。

（1）虚拟机的创建、操作系统的安装、主机名与网络的配置，具体方法参见项目一。

（2）建议通过虚拟机复制快速创建所需环境。通过复制创建的虚拟机需依据本任务虚拟机与操作系统规划配置信息设置主机名与网络，实现对互联网和本地主机的访问。

步骤 2：部署 Docker。

通过 yum 工具在线安装 Docker，本任务使用版本为 18.09.0。

操作命令：

```
1.  # 通过 yum 安装 Docker
2.  [root@Project-07-Task-05 ~]# yum -y install docker
3.  Last metadata expiration check: 0:02:16 ago on Sat 14 Oct 2023 12:00:12 AM CST.
4.  Dependencies resolved.
5.  ================================================================================
6.   Package          Architecture    Version                    Repository    Size
7.  ================================================================================
```

```
Installing:
 docker-engine     x86_64          2:18.09.0-323.oe2203sp2    update          39 M
Installing dependencies:
 Libcgroup         x86_64          2.0.3-1.oe2203sp2          update          100 k
 libtool-ltdl      x86_64          2.4.7-1.oe2203sp2          OS              33 k
 ta                x86_64          2:1.34-4.oe2203sp2         update          785 k

Transaction Summary
================================================================================
Install  4 Packages

Total download size: 40 M
Installed size: 166 M
Downloading Packages:
(1/4): libcgroup-2.0.3-1.oe2203sp2.x86_64.rpm                 153 kB/s | 100 kB     00:00
(2/4): libtool-ltdl-2.4.7-1.oe2203sp2.x86_64.rpm              17 kB/s | 33 kB   00:01
(3/4): tar-1.34-4.oe2203sp2.x86_64.rpm                        105 kB/s | 785 kB     00:07
(4/4): docker-engine-18.09.0-323.oe2203sp2.x86_64.rpm         113 kB/s | 39 MB05:55
--------------------------------------------------------------------------------
Total                                                         116 kB/s | 40 MB05:55
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                                                                1/1

  Verifying        : tar-2:1.34-4.oe2203sp2.x86_64                         4/4
# 为了排版方便此处省略了部分提示信息
Installed:
  docker-engine-2:18.09.0-323.oe2203sp2.x86_64                libcgroup-2.0.3-1.oe2203sp2.x86_64
libtool-ltdl-2.4.7-1.oe2203sp2.x86_64          tar-2:1.34-4.oe2203sp2.x86_64

Complete!
# 查看 Docker 状态
[root@Project-07-Task-05 ~]# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
     Active: active (running) since Sat 2023-10-14 00:09:03 CST; 3min 28s ago
       Docs: https://docs.docker.com
   Main PID: 1616 (dockerd)
      Tasks: 15 (limit: 2703)
     Memory: 99.0M
     CGroup: /system.slice/docker.service
             ├─1616 /usr/bin/dockerd --live-restore
             └─1620 containerd --config /var/run/docker/containerd/containerd.toml --log-level info
Oct 14 00:09:00 Project-07-Task-05 dockerd[1616]: time="2023-10-14T00:09:00.375111362+08:00" level=
warning msg="Failed to cleanup netns file /var/run/docker/runtime-runc: remove /v>
Oct 14 00:09:01 Project-07-Task-05 dockerd[1616]: time="2023-10-14T00:09:01.579259958+08:00" level=
info msg="Default bridge (docker0) is assigned with an IP address 172.17.0.0/16. >
```

```
55. Oct 14 00:09:01 Project-07-Task-05 dockerd[1616]: time="2023-10-14T00:09:01.581217039+08:00" level=
    info msg="Setup IP tables begin"
56. Oct 14 00:09:01 Project-07-Task-05 dockerd[1616]: time="2023-10-14T00:09:01.945452797+08:00" level=
    info msg="Setup IP tables end"
57. Oct 14 00:09:02 Project-07-Task-05 dockerd[1616]: time="2023-10-14T00:09:02.243780083+08:00" level=
    info msg="Loading containers: done."
58. Oct 14 00:09:03 Project-07-Task-05 dockerd[1616]: time="2023-10-14T00:09:03.388814637+08:00" level=
    info msg="Docker daemon" commit=172f8da graphdriver(s)=overlay2 version=18.09.0
59. Oct 14 00:09:03 Project-07-Task-05 dockerd[1616]: time="2023-10-14T00:09:03.434029428+08:00" level=
    info msg="Daemon has completed initialization"
60. Oct 14 00:09:03 Project-07-Task-05 dockerd[1616]: time="2023-10-14T00:09:03.860488205+08:00" level=
    warning msg="Could not register builder git source: failed to find git binary: ex>
61. Oct 14 00:09:03 Project-07-Task-05 dockerd[1616]: time="2023-10-14T00:09:03.974121995+08:00" level=I
    nfo msg="API listen on /var/run/docker.sock"
62. Oct 14 00:09:03 Project-07-Task-05 systemd[1]: Started Docker Application Container Engine.
63. # 查看 Docker 版本
64. [root@Project-07-Task-05 ~]# docker -v
65. Docker version 18.09.0, build 172f8da
66. # 设置 Docker 自启动
67. [root@Project-07-Task-05 ~]# systemctl enable docker
```

操作命令+配置文件+脚本程序+结束

步骤 3：部署 PMM Server。

Percona 官方提供 percona/pmm-server 镜像，当前版本为 1.12.6，其官方网站为 https://www.percona.com/software/database-tools/percona-monitoring-and-management。

操作命令：

```
1.  # 拉取 percona/pmm-server 的 Docker 镜像
2.  [root@Project-07-Task-05 ~]# docker pull percona/pmm-server
3.  Using default tag: latest
4.  latest: Pulling from percona/pmm-server
5.  ccf3a7357f12: Pull complete
6.  16ae537e810b: Pull complete
7.  Digest: sha256:4a4ff78ae4fc42194f77d0361ba03687b24c81dfe5c1ea78bd720fc1167a0f70
8.  Status: Downloaded newer image for percona/pmm-server:latest
9.  # 为 pmm-server 创建一个永久数据容器
10. [root@Project-07-Task-05 ~]# docker create --volume /srv --name pmm-data percona/pmm-server /bin/tru
    e
11. 2a7c3487507145118559 67b7d38d95a3ec8e6252289df040e9653fb7f4050d76
12. # 运行 pmm-server 镜像
13. [root@Project-07-Task-05 ~]# docker run --detach --restart always --publish 443:443 --publish 80:80 --v
    olumes-from pmm-data --name pmm-server percona/pmm-server
14. 6f099192d1c88527e039b6aaaa4fafb93b3ba54d802abc0493e605326436dfb2
```

操作命令+配置文件+脚本程序+结束

运行 PMM Server 镜像后，需等待其自动初始化配置完成，否则将出现“500 Internal Server Error”服务器错误提示。

完成安装后，使用浏览器中访问服务器 http://10.10.2.75，进入其登录界面，如图 7-5-1 所示。

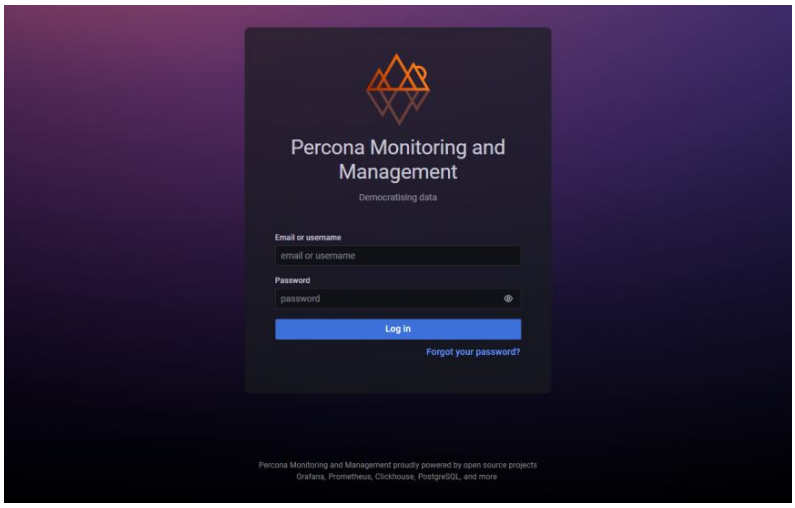

图 7-5-1　PMM Server 登录界面

初始默认用户密码为 admin/admin，首次登录系统要求修改默认密码，根据提示进行修改完成后，进入系统如图 7-5-2 所示，说明安装成功。

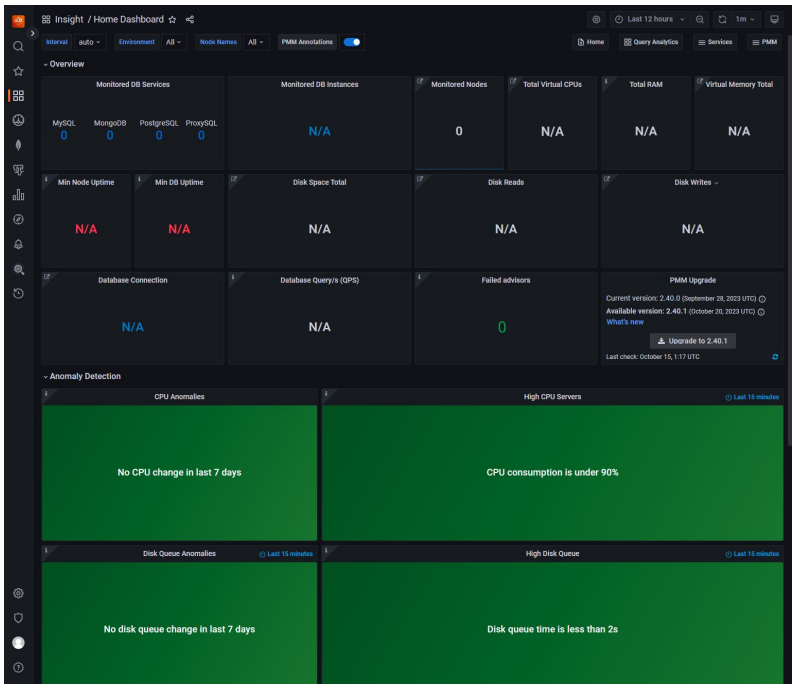

图 7-5-2　PMM Server 首页

步骤 4：部署 PMM Client。

Percona 官方提供 percona/pmm-server 镜像，当前版本为 1.12.6，安装并注册成功后，可在 PMM Server 的设置中的“Nodes”标签页中看到注册节点，如图 7-5-3 所示。

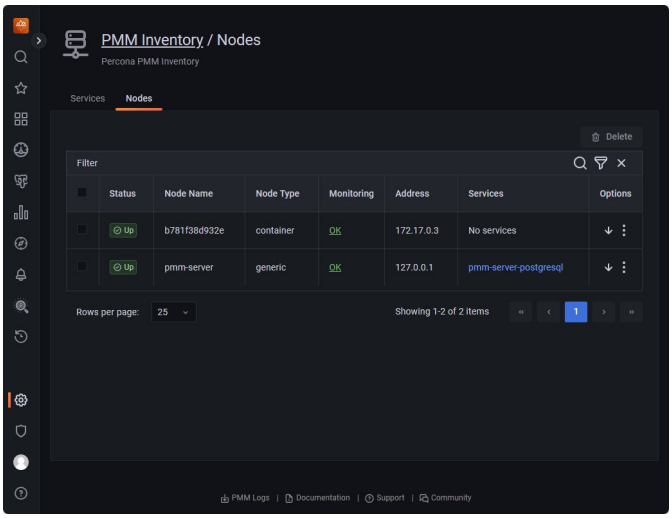

图 7-5-3　注册节点

操作命令：

```
# 拉取 percona/pmm-client 的 Docker 镜像
[root@Project-07-Task-05 ~]# docker pull percona/pmm-client
Using default tag: latest
latest: Pulling from percona/pmm-client
9e167103aefa: Pull complete
89451b1adfe4: Pull complete
Digest: sha256:5647f750d04af06460ccd27f59d4b08e3d687fd886c1de23eafa630440e4fec6
Status: Downloaded newer image for percona/pmm-client:latest
# 为 pmm-client 创建一个永久数据容器
[root@Project-07-Task-05 ~]#  docker create --volume /srv --name pmm-client-data percona/pmm-client /bin/true
f34289835c60ea69b7b12e5dbee541ec20d3dcbca1d3ed4c030020a71753d679
# 运行 pmm-client 镜像并向 pmm-server 注册 pmm-client 节点
[root@Project-07-Task-05 ~]# docker run \
  --rm \
  --name pmm-client \
  -e PMM_AGENT_SERVER_ADDRESS=10.10.2.75 \
  -e PMM_AGENT_SERVER_USERNAME=admin \
  -e PMM_AGENT_SERVER_PASSWORD=openEuler@pmm \
  -e PMM_AGENT_SERVER_INSECURE_TLS=1 \
  -e PMM_AGENT_SETUP=1 \
```

```
21.    -e PMM_AGENT_CONFIG_FILE=config/pmm-agent.yaml \
22.    --volumes-from pmm-client-data \
23.    percona/pmm-client
24.  # 当出现以下信息说明注册成功
25.  time="2023-10-14T09:26:36.516+00:00" level=info msg="Run setup: true Sidecar mode: false" component=
     entrypoint
26.  time="2023-10-14T09:26:36.517+00:00" level=info msg="Starting 'pmm-admin setup'..." component=
     entrypoint
27.  time="2023-10-14T09:26:36.560+00:00" level=info msg="Loading configuration file /usr/local/percona/
     pmm2/config/pmm-agent.yaml." component=setup
28.  time="2023-10-14T09:26:36.561+00:00" level=info msg="Temporary directory is not configured and will be
     set to /usr/local/percona/pmm2/tmp" component=setup
29.  # 为了排版方便，以下输出信息中省略了部分信息
30.  .......
31.  time="2023-10-14T09:26:36.561+00:00" level=info msg="Updating PMM Server address from \"10.10.2.75\"
     to \"10.10.2.75:443\"." component=setup
32.  Checking local pmm-agent status...
33.  pmm-agent is not running.
34.  Registering pmm-agent on PMM Server...
35.  Registered.
36.  Configuration file /usr/local/percona/pmm2/config/pmm-agent.yaml updated.
37.  Please start pmm-agent: 'pmm-agent --config-file=/usr/local/percona/pmm2/config/pmm-agent.yaml'.
38.  time="2023-10-14T09:26:36.848+00:00" level=info msg="'pmm-admin setup' exited with 0" component=
     entrypoint
39.  time="2023-10-14T09:26:36.848+00:00" level=info msg="Starting 'pmm-admin run'..." component=entrypoint
40.  time="2023-10-14T09:26:36.881+00:00" level=info msg="Loading configuration file /usr/local/percona/
     pmm2/config/pmm-agent.yaml." component=main
41.  time="2023-10-14T09:26:36.882+00:00" level=info msg="Using /usr/local/percona/pmm2/exporters/
     node_exporter" component=main
42.  # 为了排版方便，以下输出信息中省略了部分信息
43.  ......
44.  time="2023-10-14T09:26:36.883+00:00" level=info msg="Updating PMM Server address from \"10.10.2.75\"
     to \"10.10.2.75:443\"." component=main
45.  time="2023-10-14T09:26:36.883+00:00" level=info msg="Runner capacity set to 32." component=runner
46.  time="2023-10-14T09:26:36.883+00:00" level=info msg="Window check connection time is 1.00 hour(s)"
47.  time="2023-10-14T09:26:36.883+00:00" level=info msg=Starting... component=client
48.  time="2023-10-14T09:26:36.883+00:00" level=info msg="Connecting to https://admin:***@10.10.2.75:
     443/ ..." component=client
49.  time="2023-10-14T09:26:36.885+00:00" level=info msg="Starting local API server on http://127.0.0.1:
     7777/ ..." component=local-server/JSON
50.  time="2023-10-14T09:26:36.889+00:00" level=info msg=Started. component=local-server/JSON
51.  time="2023-10-14T09:26:36.895+00:00" level=info msg="Connected to 10.10.2.75:443." component=client
```

```
52. time="2023-10-14T09:26:36.895+00:00" level=info msg="Establishing two-way communication channel ..." component=client
53. time="2023-10-14T09:26:37.275+00:00" level=info msg="Two-way communication channel established in 379.951194ms. Estimated clock drift: 188.145953ms." component=client
54. time="2023-10-14T09:26:38.370+00:00" level=info msg="Starting 2, restarting 0, and stopping 0 agent processes." component=supervisor
55. time="2023-10-14T09:26:38.385+00:00" level=info msg="Sending status: STARTING (port 42000)." agentID=/agent_id/2c019ecd-4c29-4f35-bb90-8423d14176cd component=agent-process type=vm_agent
56. ......
```

操作命令+配置文件+脚本程序+结束

步骤 5：配置 MongoDB。

本任务选择监控任务三中的 MongoDB 副本集，在监控前需要对 MongoDB 进行配置，创建具有创建/还原备份和使用查询分析（QAN）所需权限的自定义角色；使用上述角色以及内置角色创建/更新数据库用户 clusterMonitor。

操作命令：

```
1.  [root@Project-07-Task-02 ~]# mongosh
2.  Current Mongosh Log ID: 651065e3cb1e8b8bbc92f796
3.  Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.9.1
4.  Using MongoDB: 6.0.8
5.  Using Mongosh: 1.9.1
6.
7.  For mongosh info see: https://docs.mongodb.com/mongodb-shell/
8.
9.  db-cluster-mongodb [direct: primary] test> use admin;
10. switched to db admin
11. db-cluster-mongodb [direct: primary] admin> db.auth("admin","openeuler@mongodb#123");
12. { ok: 1 }
13. # 创建具有备份权限的角色
14. db-cluster-mongodb [direct: primary] admin> db.getSiblingDB("admin").createRole({
15. ...     role: "explainRole",
16. ...     privileges: [{
17. ...         resource: {
18. ...             db: "",
19. ...             collection: ""
20. ...         },
21. ...         actions: [
22. ...             "listIndexes",
23. ...             "listCollections",
24. ...             "dbStats",
25. ...             "dbHash",
26. ...             "collStats",
27. ...             "find"
```

```
...                 ]
...         }],
...         roles: []
... });
{
    ok: 1,
    '$clusterTime': {
        clusterTime: Timestamp({ t: 1695573530, i: 2 }),
        signature: {
            hash: Binary(Buffer.from("946aac880d7fde127161ad67754207e3c15a5656", "hex"), 0),
            keyId: Long("7275017387823333381")
        }
    },
    operationTime: Timestamp({ t: 1695573530, i: 2 })
}
# 创建具有 QAN 权限的角色
db-cluster-mongodb [direct: primary] admin> db.getSiblingDB("admin").createRole({
... "role": "pbmAnyAction",
... "privileges": [{
... "resource": {
... "anyResource": true
... },
... "actions": [
... "anyAction"
... ]
... }],
... "roles": []
... });
{
    ok: 1,
    '$clusterTime': {
        clusterTime: Timestamp({ t: 1695573563, i: 1 }),
        signature: {
            hash: Binary(Buffer.from("5a1324ef6701ae9a807bdf0e98b1a63ecc054143", "hex"), 0),
            keyId: Long("7275017387823333381")
        }
    },
    operationTime: Timestamp({ t: 1695573563, i: 1 })
}
# 创建 pmm 用户并分配上面创建的角色
db-cluster-mongodb [direct: primary] admin> db.getSiblingDB("admin").createUser({
...         user: "pmm",
...         pwd: "pmm",
...         roles: [
...                 { role: "explainRole", db: "admin" },
...                 { role: "clusterMonitor", db: "admin" },
...                 { role: "read", db: "local" },
...                 { "db" : "admin", "role" : "readWrite", "collection": "" },
...                 { "db" : "admin", "role" : "backup" },
...                 { "db" : "admin", "role" : "clusterMonitor" },
```

```
78.  ...                { "db" : "admin", "role" : "restore" },
79.  ...                { "db" : "admin", "role" : "pbmAnyAction" }
80.  ...            ]
81.  ... });
82.  {
83.      ok: 1,
84.      '$clusterTime': {
85.        clusterTime: Timestamp({ t: 1695573581, i: 1 }),
86.        signature: {
87.          hash: Binary(Buffer.from("a02abdf4d6ab164ea3bcce6aaf40226679b9d27a", "hex"), 0),
88.          keyId: Long("7275017387823333381")
89.        }
90.      },
91.      operationTime: Timestamp({ t: 1695573581, i: 1 })
92.  }
```

操作命令+配置文件+脚本程序+结束

步骤 6：添加 Service。

配置数据库服务器后，可以使用用户界面或在命令行上添加 MongoDB 服务。本任务选择前者。在 PMM Server 的设置中选择“+Add Service”，选择“MongoDB”添加 MongoDB 服务。第一步选择数据库服务类型，如图 7-5-4 所示；第二步配置 MongoDB 服务信息，如图 7-5-5 所示。添加成功后可在左侧导航中看到 MongoDB 菜单。

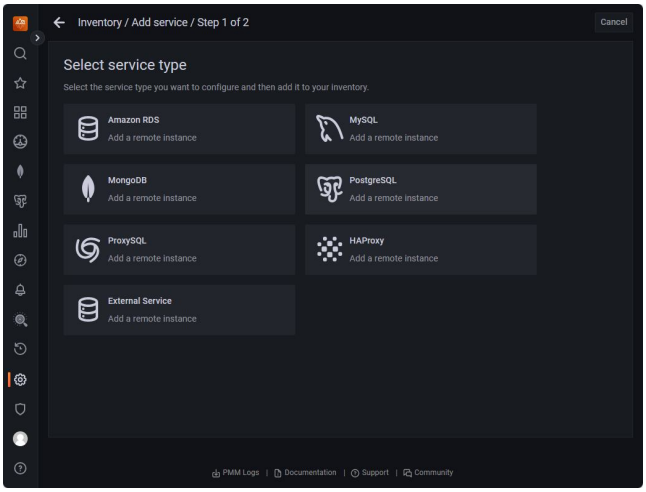

图 7-5-4　选择服务类型

步骤 7：查看 PMM 监控。

1. Overview

在 PMM Server 左侧导航选择 MongoDB 项，单击“Overview”选项卡查看 MongoDB 实例的基本信息，如图 7-5-6 所示。

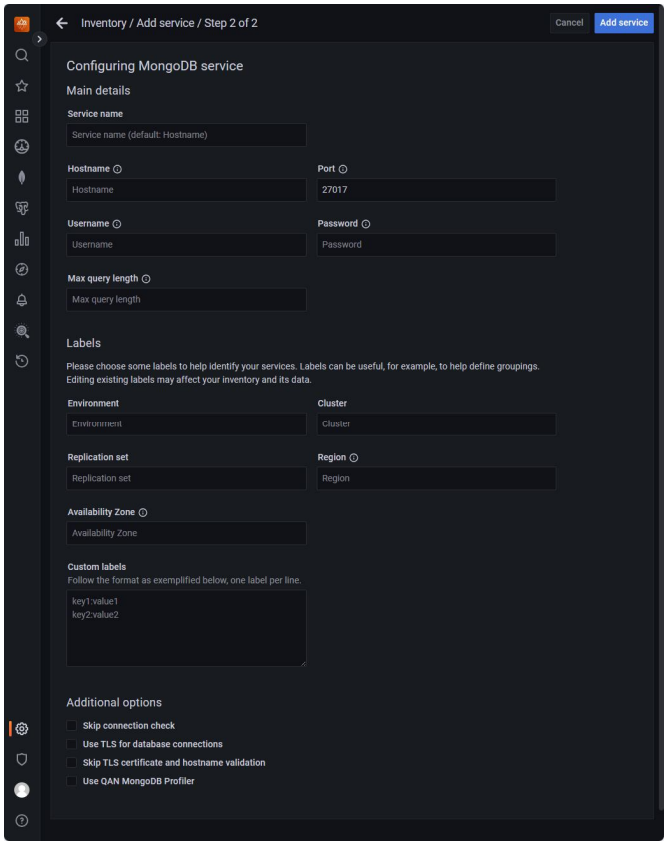

图 7-5-5　配置 MongoDB 服务

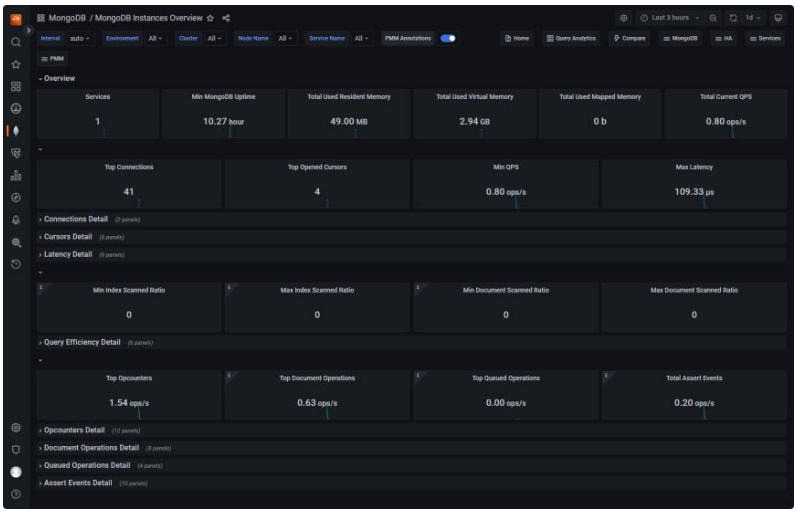

图 7-5-6　Overview

Overview 中指标说明。

- Command Operations：命令操作，显示所选时间间隔内平均每秒执行命令的次数，查找峰值和下降，并将它们与其他图形相关联
- Connections：连接，显示 mongod 进程中的 TCP 连接（传入）
- Cursors：游标，显示活动游标与由于应用程序未关闭连接而在 10min 后自动终止的游标的比较，帮助确定连接增加的原因

小贴士

- Document Operations：文档操作，显示给定时间段内平均受影响的文档的实际数量
- Queued Operations：队列操作，显示了由于锁定而等待的读取和写入操作数
- Asserts：断言，显示给定时间段内平均每秒的断言事件数
- Memory Faults：内存故障，显示内存故障表示由于缺少索引或数据集内存不足

2. Summary

在 PMM Server 左侧导航选择 MongoDB 项，单击“Summary”选项卡查看 MongoDB 实例的概要信息，如图 7-5-7 所示。

Summary 中指标说明。

- Command Operations：命令操作，按传统线路协议类型（查询、插入、更新、删除、获取更多）分类的操作数或复制操作数每秒。并且（从内部 TTL 线程）文档按 TTL 索引每秒删除
- Latency Detail：延迟详细信息，操作的平均延迟［按读取、写入或（其他）命令分类］
- Document Operations：文档操作，每秒插入、更新、删除或返回的文档数

小贴士

- Query Efficiency：查询效率，返回的文档或扫描的索引条目/扫描的完整文档的比率的信息
- Scanned and Moved Objects：扫描和移动的对象，显示对象数［数据(scanned_objects)和索引（扫描）］以及由于文档大小增大而移动到新位置的文档数
- Write Time：写时间，每秒执行命令以确认写入问题所花费的时间数和总和
- Write Operations：写操作，尝试确认写入问题时超时的命令数码
- Page Faults：页面错误，UNIX 或 Windows 内存页错误。不一定来自 MongoDB

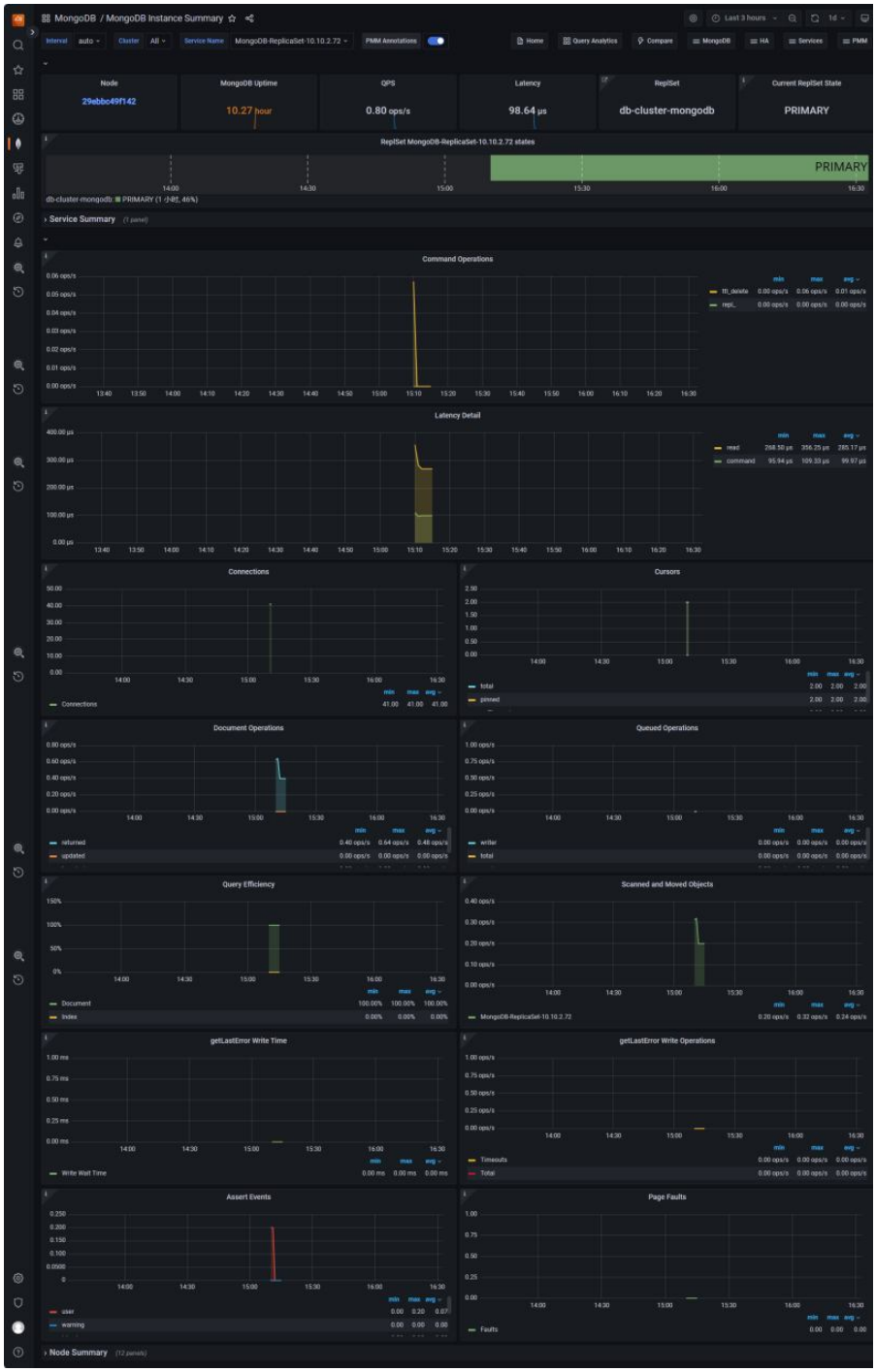

图 7-5-7　Summary

3. High availability

在 PMM Server 左侧导航选择 MongoDB 项，选择“High availability”，单击“ReplSet Summary”选项卡，查看 MongoDB ReplSet 的概要信息，如图 7-5-8 所示。

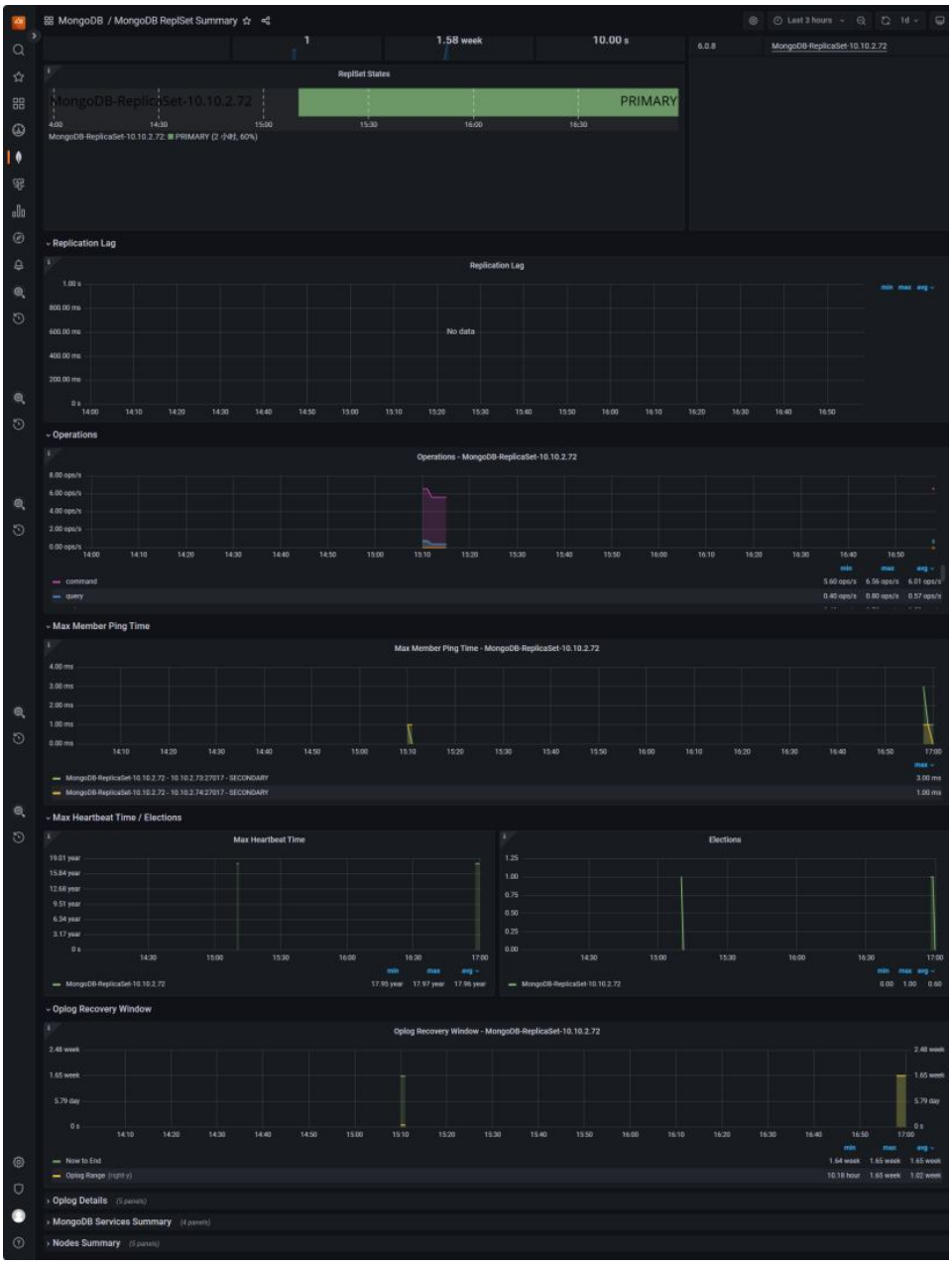

图 7-5-8 ReplSet Summary

ReplSet Summary 中指标说明。

- Replication Lag：复制滞后，当辅助节点无法以足够快的速度复制数据以跟上数据写入主节点的速率时，就会发生 MongoDB 复制滞后。它可能是由网络延迟、网络内的数据包丢失或路由问题等简单原因引起的
- Operations-byservicename：基于服务名称的操作，操作按传统线路协议类型（仅插入、更新和删除）进行分类
- Max Member Ping Time-byservicename：基于服务名称的最大成员 Ping 时间，以 ping 为衡量指标显示复制滞后值的相关服务
- Max Heartbeat Time：最大心跳时间，从现在到副本集成员的上次检测信号之间的时间跨度
- Elections：选举，显示选举计数。通常为零，每次选举中将出现每个健康节点的 1 个计数。当主要角色由于正常维护或故障事件而更改时发生
- Oplog Recovery Window-by service name：基于服务名称的 Oplog 恢复窗口，Oplog 集合中最新和最旧运算之间的时间跨度“窗口”

小贴士

4. InMemory

在 PMM Server 左侧导航选择 MongoDB 项，单击“InMemory”选项卡，查看 MongoDB ReplSet 的概要信息，如图 7-5-9 所示。

InMemory 中指标说明。

- InMemory Transactions：内存事务，显示 WiredTiger 内部交换
- InMemory Capacity：内存容量，配置了 WiredTiger 缓存的最大值和当前值
- InMemory Sessions：内存会话，内部 WiredTiger 存储引擎游标和会话当前处于打开状态
- InMemory Pages：内存页面，WiredTiger 缓存中的页面
- InMemory Concurrency Tickets：内存并发票证，在 WiredTiger 存储引擎中同时运行的每个操作都会分配一个 WiredTiger“票证”，“可用门票”=硬编码的高价值—“门票售罄”
- Document Changes：文档更改，混合指标：在任何类型的节点（主节点或辅助节点）上每秒插入、更新、删除或返回的文档数，其包括复制写入操作数和 TTL 每秒删除次数
- InMemory Cache Eviction：内存中缓存逐出，显示给定时间段内已从 WiredTiger 缓存中逐出的页数。InMemory 存储引擎仅逐出已修改的页面，这表示压缩数据并删除脏页面

小贴士

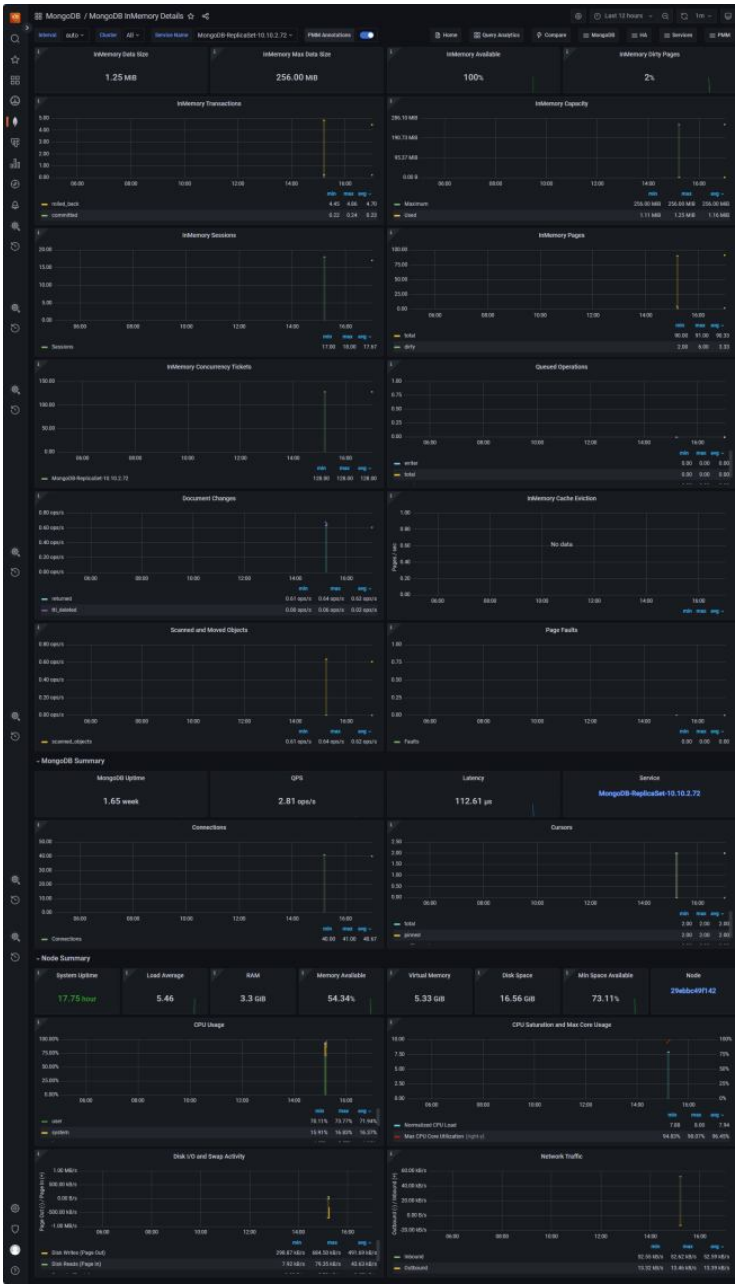

图 7-5-9　InMemory

5. MMAPv1

在 PMM Server 左侧导航选择 MongoDB 项，单击“MMAPv1”选项卡，查看 MongoDB MMAPv1 的概要信息，如图 7-5-10 所示。

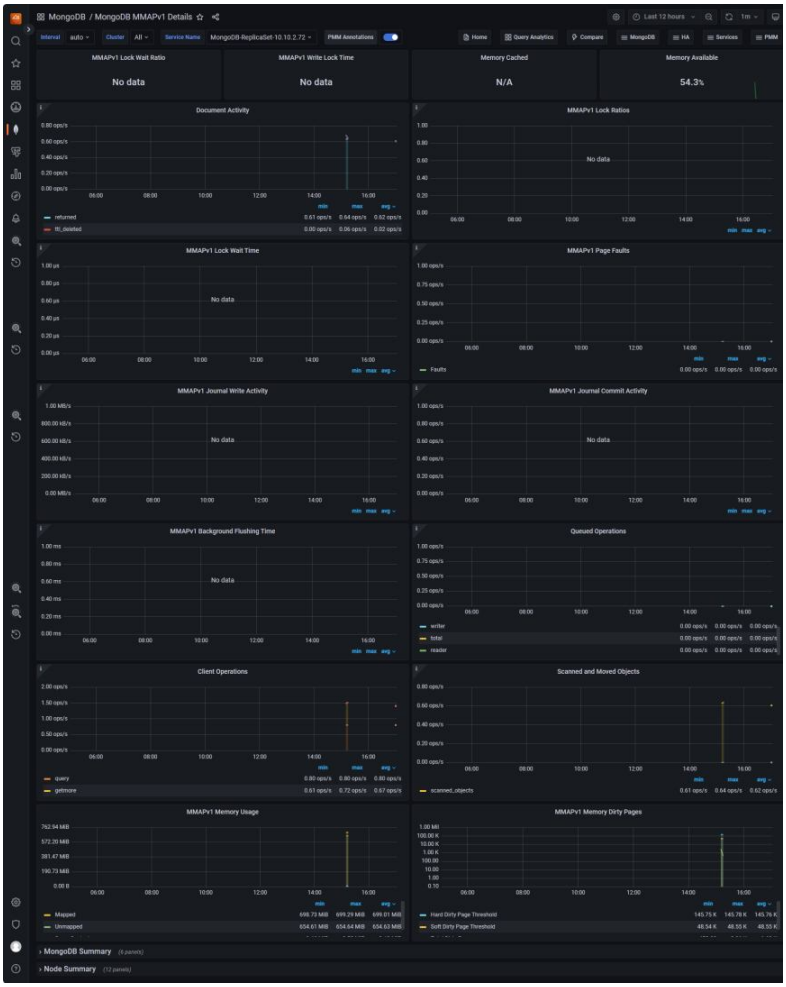

图 7-5-10　MMAPv1

MMAPv1 中指标说明。

- Document Activity：文档活动，显示每秒插入、更新、删除或返回的文档数和复制的写入操作和内部 TTL 索引删除
- MMAPv1 Lock Wait Time：锁定等待时间，显示每秒等待获取锁所花费的时间
- MMAPv1 Journal Write Activity：日志写入活动，显示 MongoDB 通过内存中的日志进行处理
- MMAPv1 Journal Commit Activity：日志提交活动，已提交到日志磁盘的 MongoDB 数据
- MMAPv1 Background Flushing Time：后台刷新时间，在 mongod 进程的完

小贴士

整正常运行时间内，MMAP 后台刷新已花费的平均时间（单位：ms）

- Queued Operations：队列操作，显示等待提交到存储引擎层的运算的队列大小
- Client Operations：客户端操作，操作数和复制操作数/秒，按传统线路协议类型（查询、插入、更新、删除、获取更多）分类

6. WiredTiger

在 PMM Server 左侧导航选择 MongoDB 项，单击“InMemory”选项卡，查看 MongoDB ReplSet 的概要信息，如图 7-5-11 所示。

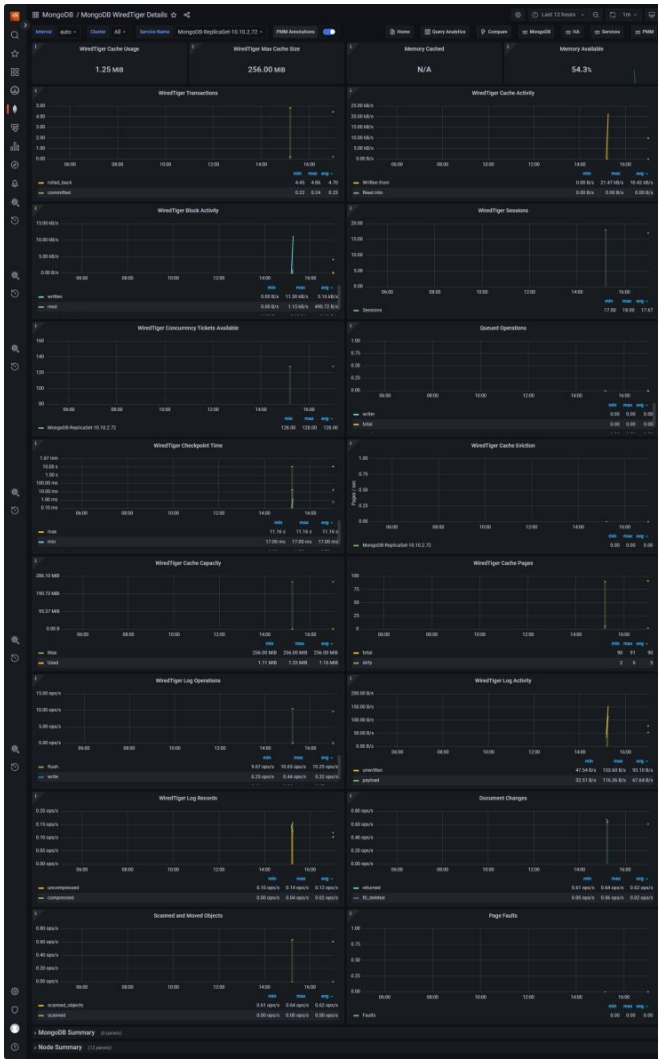

图 7-5-11　WiredTiger

WiredTiger（以下简称“WT”）中指标说明。

- WiredTiger Transactions：交换，显示 WT 内部交易
- WiredTiger Cache Activity：缓存活动，即每秒在 WT 缓存与数据文件之间传输的数据量。写操作始终意味着数据写入磁盘；读取操作通常从操作系统文件缓冲区缓存中的 RAM 获取数据，但如果缓存中没有相应数据，则从磁盘读取
- WiredTiger Block Activity：区块活动，WT 块管理器每秒处理的数据量
- WiredTiger Sessions：会话，当前打开的内部 WT 存储引擎游标和会话
- WiredTiger ConcurrencyTickets Available：并发票可用性，在 WT 存储引擎中，每个同时运行的操作都会分配一个 WT“票证”。“可用”票证的数量等于硬编码的最大值减去当前已分配的票证数。
- WiredTiger Checkpoint Time：检查点时间，在 WT 检查点阶段花费的时间
- WiredTiger Cache Eviction：缓存逐出，由于 WT 缓存已满，最近使用最少的页面被逐出
- WiredTiger Cache Capacity：缓存容量，显示配置 WT 缓存的最大值和当前值
- WiredTiger Log Operations：日志操作，显示 WT 内部预写日志操作
- WiredTiger Log Activity：日志活动，显示 WT 内部预写日志中每秒移动的数据量
- WiredTiger Log Records：日志记录，显示 WT 内部日志中每秒追加的记录数

小贴士