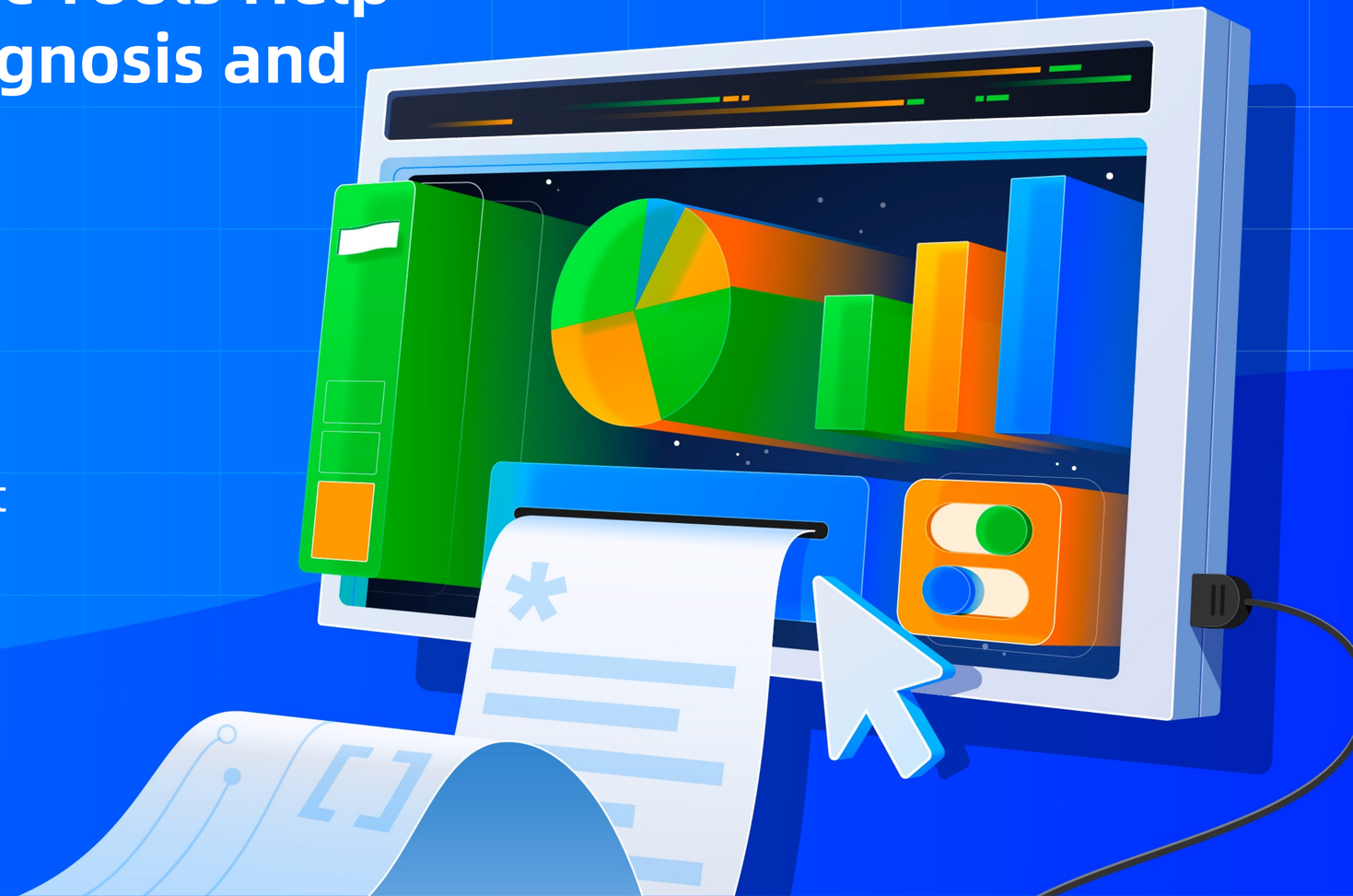
FROM INTRODUCTION TO PRACTICE

Lesson 9: Agile Diagnostic Tools Help Oceanbase Database Diagnosis and Tuning

Peng Wang

OceanBase Global Technical Evangelist

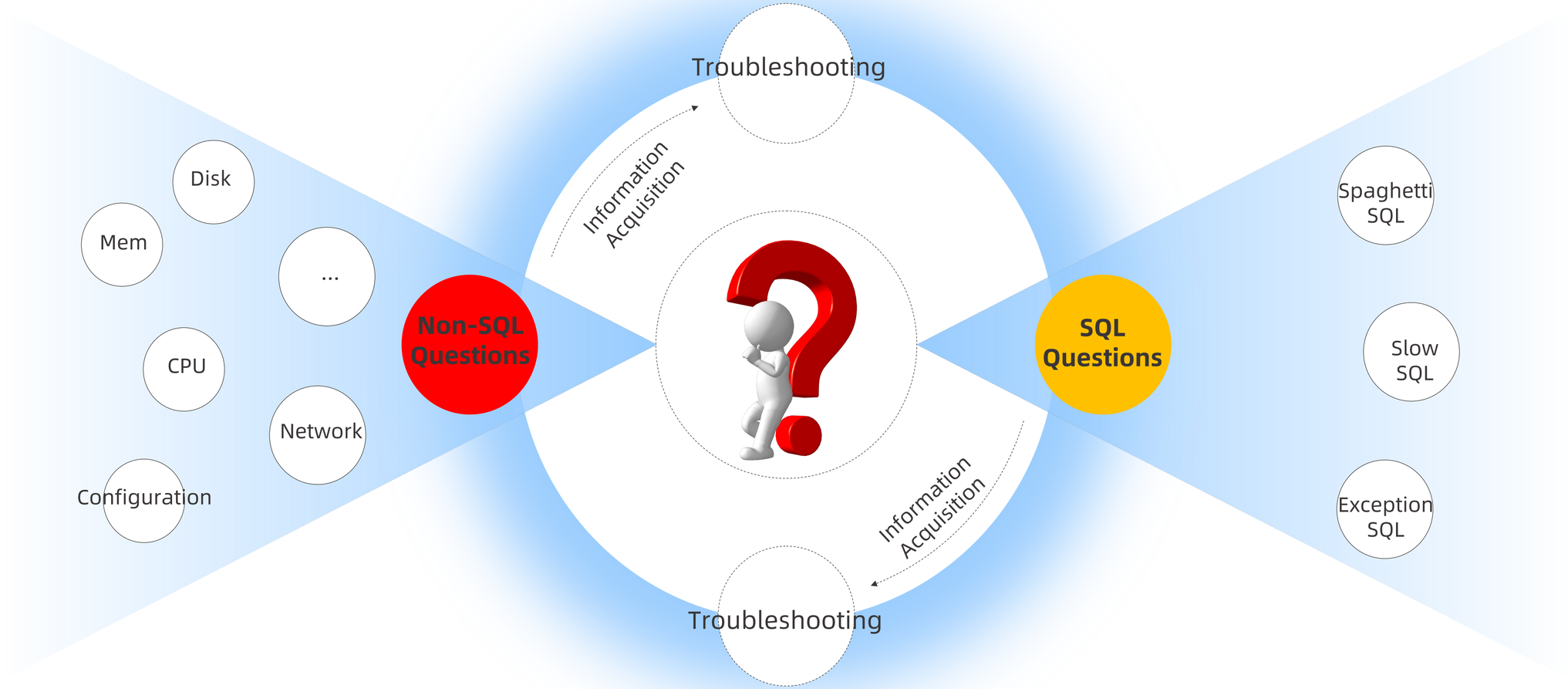


Agenda

- Introduction to Agile Diagnostic Tools
- Use the Obdiag Tool for Self-diagnosis
- Add Obdiag Diagnostic Scenario by Yourself
- Agile Diagnostic Tools Summary and Outlook

Introduction to Agile Diagnostic Tools

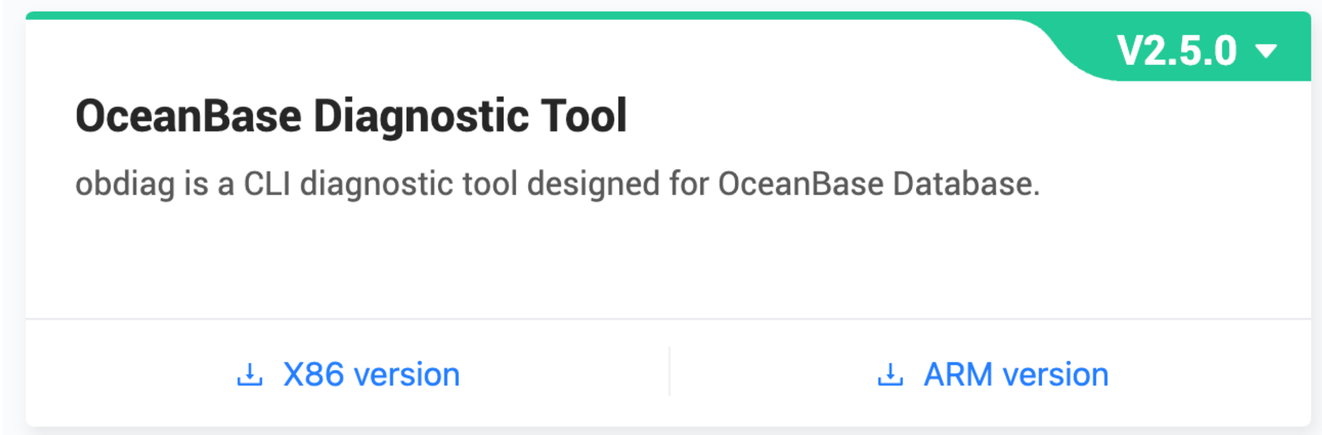
OceanBase Database Problem Classification



Introduction to Agile Diagnostic Tool

Product Introduction

https://en.oceanbase.com/softwarecenter



OceanBase Diagnostic Tool (obdiag) is an open-source agile CLI diagnostic tool.

The existing functions of obdiag include scanning, collection, and analysis of OceanBase database logs, SQL Audit, and OceanBase database process stack information. It can be executed with one click in different deployment modes of the OceanBase cluster (OCP, obd, or manual deployment by the user according to the documentation) to complete the collection and analysis of diagnostic information.

Agenda

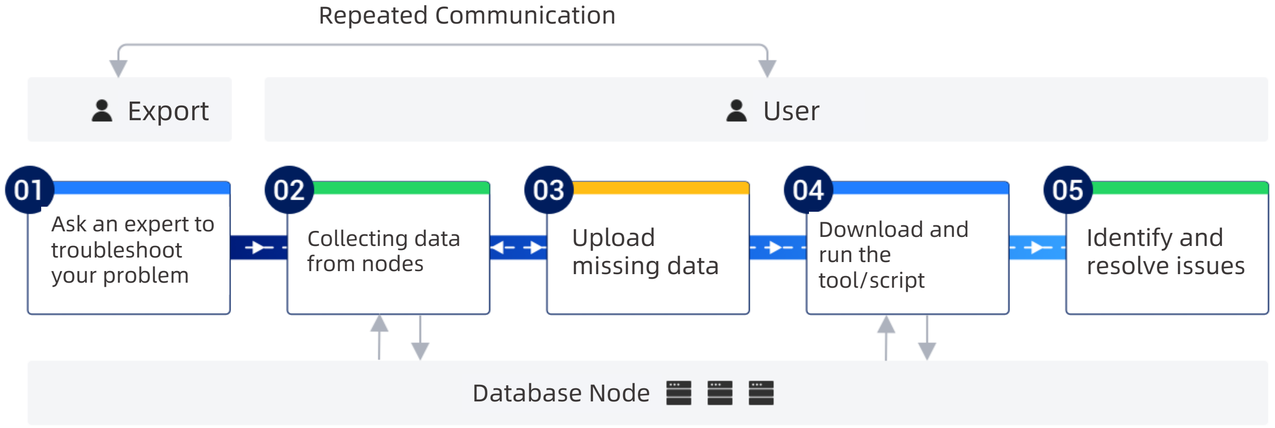
- Introduction to Agile Diagnostic Tools
- Use the Obdiag Tool for Self-diagnosis
- Add Obdiag Diagnostic Scenario by Yourself
- Agile Diagnostic Tools Summary and Outlook

Self-service Diagnosis Using Obdiag Tool

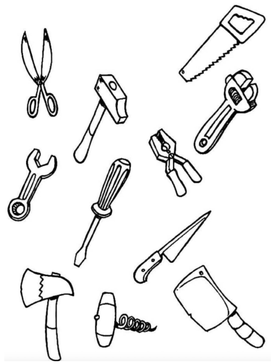
FROM INTRODUCTION TO PRACTICE

Introduction to obdiag Tool

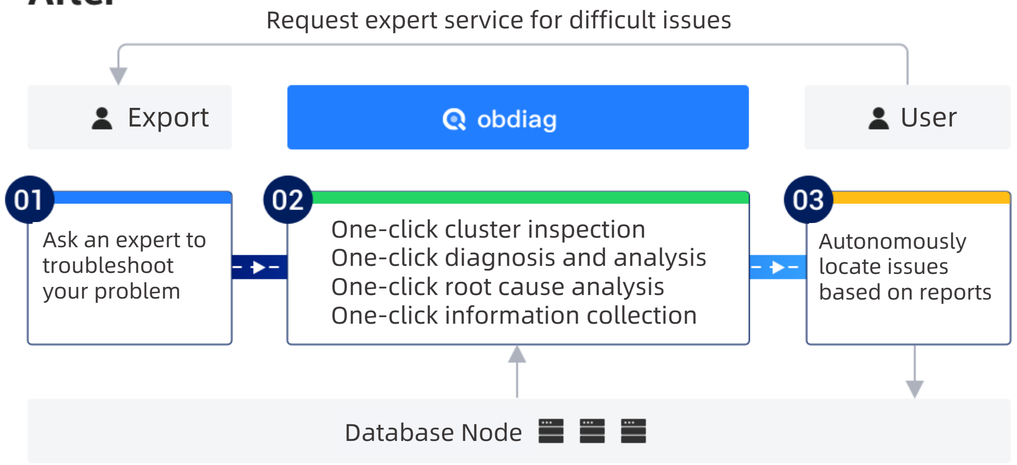
Before



Before =



After



After =

Self-service Diagnosis Using Obdiag Tool

Introduction to obdiag Features

OceanBase Diagnostic Tool (obdiag) is an open-source agile CLI diagnostic tool that can perform one-click cluster inspection, one-click analysis, and one-click diagnostic information collection on the OceanBase cluster.

Extremely Lightweight

A 30MB-sized one-click deployment tool that can be used out of the box

Easy to Use

One-click deployment
One-click execution

Fully Open Source

Source code open source

Highly Scalable

Plug-in scenarios
Highly scalable

obdiag has four core functions: one-click cluster inspection, one-click information collection, one-click analysis, and one-click root cause analysis.

One-click Cluster Inspection

Supports one-click cluster health inspection for more than 30 inspection items

One-click Analysis

One-click log analysis
One-click full-link analysis

One-click Root Cause Analysis

One-click root cause analysis

One-click Information Collection

Support one-click information collection for 20+ fault scenarios

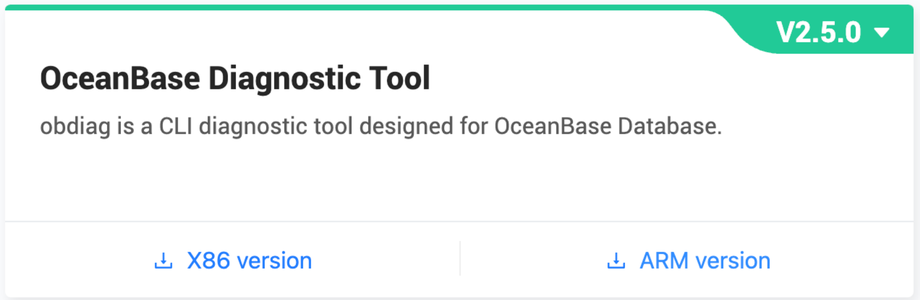
Use of obdiag Tool for Self-diagnosis

Installation and Deployment

Method 1: Install obdiag online independently
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://mirrors.aliyun.com/oceanbase/OceanBase.repo

```
yum install -y oceanbase-diagnostic-tool
```

Method 2: Offline independent installation of obdiag
Download: https://www.oceanbase.com/softwarecenter




```
yum install -y oceanbase-diagnostic-tool*.rpm
```

Method 3: Install obdiag using obd method

```
obd obdiag deploy
```

Use of obdiag Tool for Self-diagnosis

FROM INTRODUCTION TO PRACTICE

Configuration

User-side configuration default location: ~/.obdiag/config.yml:

Command: obdiag config -h <db_host> -u <sys_user> [-p password] [-P port]

```
#obdiag config –hxx.xx.xx.xx -Pxxxx -uroot -pxxxxx
Please enter the following configuration !!!
Enter your oceanbase host ssh username (default:’’): xxxx
Enter your oceanbase host ssh password (default:''): *********
Enter your oceanbase host ssh_port (default:'22'): 22
Enter your oceanbase install home_path (default:'/root/observer'): /home/admin/observer
Enter your oceanbase data_dir (default:'/home/admin/observer/store'):
Enter your oceanbase redo_dir (default:'/home/admin/observer/store'):
Enter your need config obproxy [y/N] (default:'N'): y
Enter your obproxy server eg:'192.168.1.1;192.168.1.2;192.168.1.3' (default:’’): xx.xx.xx.xx
Enter your obproxy host ssh username (default:’’): xxxx
Enter your obproxy host ssh password (default:''): *********
Enter your obproxy host ssh port (default:'22'): 22
Enter your obproxy install home_path (default:'/root/obproxy'): /home/admin/obproxy
Trace ID: e8c6f83c-039f-11ef-ae62-16c3aeac6f4f
If you want to view detailed obdiag logs, please run: obdiag display-trace e8c6f83c-039f-11ef-ae62-16c3aeac6f4f
```

For clusters deployed by obd, there is no need to generate additional configuration files when using obdiag. You can directly use the function. obd will be responsible for generating the configuration of obdiag.

System Configuration:
`/usr/local/oceanbase-diagnostic-tool/conf/inner_config.yml`

How to manage multiple clusters:
Generate multiple cluster configurations. When using obdiag, simply specify the corresponding cluster configuration, for example

- obdiag gather log -c cluster_1.yaml
- obdiag gather log -c cluster_2.yaml

Use the obdiag Tool for Self-diagnosis

FROM INTRODUCTION TO PRACTICE

One-click Cluster Inspection

Check Inspection Packages:
obdiag check list

check cases about observer:

Command	Description
obdiag check	default check all task without filter
obdiag check --cases=ad	Test and inspection tasks
obdiag check --cases=build_before	Deployment environment check
obdiag check --cases=sysbench_run	Collection of inspection tasks when executing sysbench
obdiag check --cases=sysbench_free	Collection of inspection tasks before executing sysbench

check cases about obproxy:

Command	Description
obdiag check	default check all task without filter
obdiag check --obproxy_cases=proxy	obproxy version check

Full Inspection:
obdiag check

Inspection before sysbench stress testing:
obdiag check --cases=sysbench_free

Inspection during sysbench stress testing:
obdiag check --cases=sysbench_run

obproxy inspection:
obdiag check -obproxy_cases=proxy

Self-service Diagnosis Using Obdiag Tool

FROM INTRODUCTION TO PRACTICE

One-click Diagnosis and Analysis

Online Analysis

One-click log analysis (30 minutes by default):
obdiag analyze log

Analyze the logs of the last 10 minutes online. When this command is executed, it will pull the logs of the last 10 minutes from the remote host for analysis and diagnose the errors that have occurred
obdiag analyze log --since 10m

One-click log analysis of logs in a specified time interval:
obdiag analyze log --from "2023-10-08 10:25:00" --to "2023-10-08 11:30:00"

Offline Analysis

ls -lh test/
-rw-r--r-- 1 admin staff 256M Oct 8 17:24 observer.log.20231008104204260
-rw-r--r-- 1 admin staff 256M Oct 8 17:24 observer.log.20231008111305072
-rw-r--r-- 1 admin staff 256M Oct 8 17:24 observer.log.20231008114410668
-rw-r--r-- 1 admin staff 18K Oct 8 17:24 observer.log.wf.20231008104204260
-rw-r--r-- 1 admin staff 19K Oct 8 17:24 observer.log.wf.20231008111305072
-rw-r--r-- 1 admin staff 18K Oct 8 17:24 observer.log.wf.20231008114410668

One-click offline analysis of all observer logs in a local folder:
obdiag analyze log --files test/

One-click offline analysis of the log of a specified file in the local:
obdiag analyze log --files test/observer.log.20231008104204260

obdiag analyze log

Analyze OceanBase Online Log Summary:

Node	Status	FileName	ErrorCode	Message	Count
192.168.1.1	PASS				
192.168.1.2	PASS				
192.168.1.3	PASS				

Self-service Diagnosis Using obdiag Tool

One-click Diagnosis and Analysis

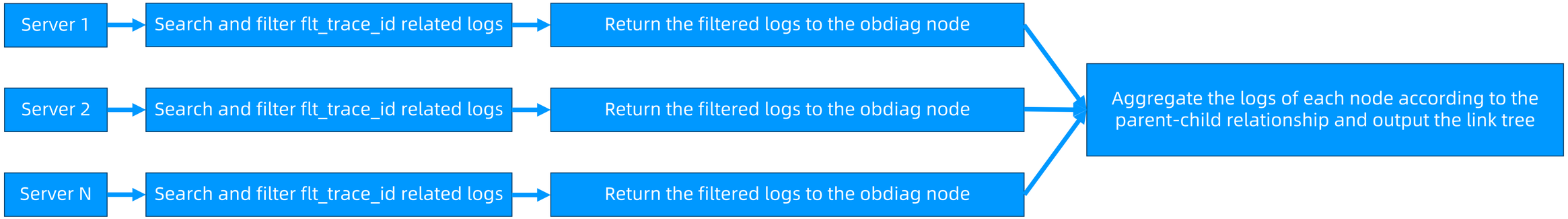
```
obdiag analyze log
analyze_log start ...
analyze log from_time: 2024-04-26 15:27:24, to_time: 2024-04-26 15:58:24

FileListInfo:
+---------------+---------------------------------------------------------------------------------------------------------+
| Node          | LogList                                                                                                 |
+===============+=========================================================================================================+
| 192.168.1.1   | ['election.log', 'observer.log', 'observer.log.20240426153336722', 'observer.log.wf', 'rootservice.log'] |
+---------------+---------------------------------------------------------------------------------------------------------+
Download 192.168.1.1:/tmp/ob_log_192.168.1.1_20240426152724_20240426155824/election.log
Downloading [====================] 100.0% [171.93 MB ]
Download 192.168.1.1:/tmp/ob_log_192.168.1.1_20240426152724_20240426155824/observer.log
Downloading [====================] 100.0% [94.99 MB ]
Download 192.168.1.1:/home/jingshun.tq/observer/log/observer.log.20240426153336722
Downloading [====================] 100.0% [256.00 MB ]
Download 192.168.1.1:/tmp/ob_log_192.168.1.1_20240426152724_20240426155824/observer.log.wf
Downloading [====================] 100.0% [109.48 KB ]
Download 192.168.1.1:/tmp/ob_log_192.168.1.1_20240426152724_20240426155824/rootservice.log
Downloading [====================] 100.0% [51.77 MB ]

Analyze OceanBase Online Log Summary:

+-------------+-----------+-----------------------------------------------------------------+-----------+--------------------------------+-------+
| Node        | Status    | FileName                                                        | ErrorCode | Message                        | Count |
+=============+===========+=================================================================+===========+================================+=======+
| 192.168.1.1 | Completed | /home/admin/analyze_pack_20240426160201/192_168_1_1/observer.log |   -4002   | Invalid argument               | 2231  |
+-------------+-----------+-----------------------------------------------------------------+-----------+--------------------------------+-------+
| 192.168.1.1 | Completed | /home/admin/analyze_pack_20240426160201/192_168_1_1/observer.log |   -4006   | The object is not initialized  |  56.  |
+-------------+-----------+-----------------------------------------------------------------+-----------+--------------------------------+-------+
For more details, please run cmd ' cat /home/admin/analyze_pack_20240426160201/result_details.txt '
```

Self-service Diagnosis Practice of obdiag Tool

One-click Full-link Diagnosis and Analysis

One-click full-link diagnostic command:

```
obdiag analyze flt_trace [options]
```



One-click full-link diagnosis example:

Step 1: In gv$ob_sql_audit, find the flt_trace_id of the sql, for example:

```
OceanBase(root@test)>select query_sql, flt_trace_id from oceanbase.gv$ob_sql_audit where query_sql like 'select
@@version_comment limit 1';
+---------------------------------+--------------------------------------+
| query_sql                       | flt_trace_id                         |
+---------------------------------+--------------------------------------+
| select @@version_comment limit 1 | 00060aa3-d607-f5f2-328b-388e17f687cb |
+---------------------------------+--------------------------------------+
1 row in set (0.001 sec)
```

Step 2: Analyze the log corresponding to this flt_trace_id, for example:

```
obdiag analyze flt_trace --flt_trace_id 000605b1-28bb-c15f-8ba0-1206bcc08aa3
```

Self-service Diagnosis Practice Of obdiag Tool

FROM INTRODUCTION TO PRACTICE

Full-link Diagnosis Analysis Report

```
$ obdiag analyze flt_trace --flt_trace_id 000605b1-28bb-c15f-8ba0-1206bcc08aa3

root node id: 000605b1-28bb-c15f-8ba0-1206bcc08aa3

TOP time-consuming leaf span:
+---+-----------------------------+------------+--------------------+
| ID| Span Name                   | Elapsed Time|      NODE          |
+---+-----------------------------+------------+--------------------+
| 18| px_task                     | 2.758 ms    | OBSERVER(xx.xx.xx.1)|
| 5 | pc_get_plan                 | 52 μs       | OBSERVER(xx.xx.xx.1)|
| 16| do_local_das_task           | 45 μs       | OBSERVER(xx.xx.xx.1)|
| 10| do_local_das_task           | 17 μs       | OBSERVER(xx.xx.xx.1)|
| 17| close_das_task              | 14 μs       | OBSERVER(xx.xx.xx.1)|
+---+-----------------------------+------------+--------------------+
Tags & Logs:
-----------------------------------
18 - px_task  Elapsed: 2.758 ms
     NODE:OBSERVER(xx.xx.xx.1)
     tags: [{'group_id': 0}, {'qc_id': 1}, {'sqc_id': 0}, {'dfo_id': 1}, {'task_id': 1}]
5 - pc_get_plan  Elapsed: 52 μs
    NODE:OBSERVER(xx.xx.xx.1)
16 - do_local_das_task  Elapsed: 45 μs
     NODE:OBSERVER(xx.xx.xx.3)
10 - do_local_das_task  Elapsed: 17 μs
     NODE:OBSERVER(xx.xx.xx.1)
17 - close_das_task  Elapsed: 14 μs
     NODE:OBSERVER(xx.xx.xx.3)

Details:
+---+-----------------------------+------------+--------------------+
| ID| Span Name                   | Elapsed Time|  NODE              |
+---+-----------------------------+------------+--------------------+
| 1 | TRACE                       | -           | -                  |
| 2 | └com_query_process          | 5.351 ms    | OBPROXY(xx.xx.xx.1) |
| 3 |   └mpquery_single_stmt      | 5.333 ms    | OBSERVER(xx.xx.xx.1)|
| 4 |     ├sql_compile            | 107 μs      | OBSERVER(xx.xx.xx.1)|
| 5 |     | └pc_get_plan          | 52 μs       | OBSERVER(xx.xx.xx.1)|
| 6 |     └sql_execute            | 5.147 ms    | OBSERVER(xx.xx.xx.1)|
| 7 |       ├open                 | 87 μs       | OBSERVER(xx.xx.xx.1)|
| 8 |       ├response_result      | 4.945 ms    | OBSERVER(xx.xx.xx.1)|
| 9 |       | ├px_schedule        | 2.465 ms    | OBSERVER(xx.xx.xx.1)|
| 10|       | | ├do_local_das_task| 17 μs       | OBSERVER(xx.xx.xx.1)|
| 11|       | | ├px_task          | 2.339 ms    | OBSERVER(xx.xx.xx.2)|
| 12|       | | | ├do_local_das_task| 54 μs       | OBSERVER(xx.xx.xx.2)|
| 13|       | | | └close_das_task | 22 μs       | OBSERVER(xx.xx.xx.2)|
| 14|       | | ├do_local_das_task| 11 μs       | OBSERVER(xx.xx.xx.1)|
| 15|       | | ├px_task          | 2.834 ms    | OBSERVER(xx.xx.xx.3)|
| 16|       | | | ├do_local_das_task| 45 μs       | OBSERVER(xx.xx.xx.3)|
| 17|       | | | └close_das_task | 14 μs       | OBSERVER(xx.xx.xx.3)|
| 18|       | | └px_task          | 2.758 ms    | OBSERVER(xx.xx.xx.1)|
| 19|       | ├px_schedule        | 1 μs        | OBSERVER(xx.xx.xx.1)|
| 20|       | └px_schedule        | 1 μs        | OBSERVER(xx.xx.xx.1)|
| ..|       ......              | ...         |  ......            |
+---+-----------------------------+------------+--------------------+

For more details, please run cmd ' cat analyze_flt_result/000605b1-28bb-c15f-8ba0-1206bcc08aa3.txt '
```

```
Details:

+---+-----------------------------+------------+--------------------+
| ID| Span Name                   | Elapsed Time|  NODE              |
+---+-----------------------------+------------+--------------------+
| 1 | TRACE                       | -           | -                  |
| 2 | └com_query_process          | 5.351 ms    | OBPROXY(xx.xx.xx.1) |
| 3 |   └mpquery_single_stmt      | 5.333 ms    | OBSERVER(xx.xx.xx.1)|
| 4 |     ├sql_compile            | 107 μs      | OBSERVER(xx.xx.xx.1)|
| 5 |     | └pc_get_plan          | 52 μs       | OBSERVER(xx.xx.xx.1)|
| 6 |     └sql_execute            | 5.147 ms    | OBSERVER(xx.xx.xx.1)|
| 7 |       ├open                 | 87 μs       | OBSERVER(xx.xx.xx.1)|
| 8 |       ├response_result      | 4.945 ms    | OBSERVER(xx.xx.xx.1)|
| 9 |       | ├px_schedule        | 2.465 ms    | OBSERVER(xx.xx.xx.1)|
| 10|       | | ├do_local_das_task| 17 μs       | OBSERVER(xx.xx.xx.1)|
| 11|       | | ├px_task          | 2.339 ms    | OBSERVER(xx.xx.xx.2)|
| 12|       | | | ├do_local_das_task| 54 μs       | OBSERVER(xx.xx.xx.2)|
| 13|       | | | └close_das_task | 22 μs       | OBSERVER(xx.xx.xx.2)|
| 14|       | | ├do_local_das_task| 11 μs       | OBSERVER(xx.xx.xx.1)|
| 15|       | | ├px_task          | 2.834 ms    | OBSERVER(xx.xx.xx.3)|
| 16|       | | | ├do_local_das_task| 45 μs       | OBSERVER(xx.xx.xx.3)|
| 17|       | | | └close_das_task | 14 μs       | OBSERVER(xx.xx.xx.3)|
| 18|       | | └px_task          | 2.758 ms    | OBSERVER(xx.xx.xx.1)|
| 19|       | ├px_schedule        | 1 μs        | OBSERVER(xx.xx.xx.1)|
| 20|       | └px_schedule        | 1 μs        | OBSERVER(xx.xx.xx.1)|
| 21|       └close                | 70 μs       | OBSERVER(xx.xx.xx.1)|
| 22|         └end_transaction    | 3 μs        | OBSERVER(xx.xx.xx.1)|
+---+-----------------------------+------------+--------------------+
Tags & Logs:
-----------------------------------
1 -
2 - com_query_process  Elapsed: 5.351 ms
    NODE:OBPROXY(xx.xx.xx.1)
    tags: [{'sess_id': 3221487633}, {'action_name': ''}, {'module_name': ''}, {'client_info': ''}, {'receive_ts': 1695108311007659},
    {'log_trace_id': 'YA9257F000001-000605B0441954BC-0-0'}]
3 - mpquery_single_stmt  Elapsed: 5.333 ms
    NODE:OBSERVER(xx.xx.xx.1)
4 - sql_compile  Elapsed: 107 μs
    NODE:OBSERVER(xx.xx.xx.1)
    tags: [{'sql_text': 'select /*+parallel(2)*/ count(1) from t1 tt1, t1 tt2'}, {'sql_id': '797B7202BA69D4C2C77C12BFADDC19DC'},
    {'database_id': 201001}, {'plan_hash': 150629045171310866}, {'hit_plan': True}]
5 - pc_get_plan  Elapsed: 52 μs
    NODE:OBSERVER(xx.xx.xx.1)
6 - sql_execute  Elapsed: 5.147 ms
    NODE:OBSERVER(xx.xx.xx.1)
7 - open  Elapsed: 87 μs
    NODE:OBSERVER(xx.xx.xx.1)
8 - response_result  Elapsed: 4.945 ms
    NODE:OBSERVER(xx.xx.xx.1)
9 - px_schedule  Elapsed: 2.465 ms
    NODE:OBSERVER(xx.xx.xx.1)
    tags: [{'used_worker_cnt': 0}, {'qc_id': 1}, {'dfo_id': 2147483647}, {'used_worker_cnt': 0}, {'qc_id': 1}, {'dfo_id': 1}]
10 - do_local_das_task  Elapsed: 17 μs
     NODE:OBSERVER(xx.xx.xx.1)
```

Self-diagnosis Practice of obdiag Tool

One-click Root Cause Analysis

FROM INTRODUCTION TO PRACTICE

One-click root cause analysis supports the following scenarios:
`obdiag rca list`

One-click hot update scenario:
`obdiag update`

Run the root cause analysis scenario with one click:
`obdiag rca run --scene=<scene_name>`

#obdiag rca list

Command	Description
obdiag rca run --scene=disconnection	root cause analysis of disconnection
obdiag rca run --scene=lock_conflict	root cause analysis of lock conflict
obdiag rca run --scene=major_hold	root cause analysis of major hold

#obdiag update
update start ...
[update] Successfully updated. The original data is stored in the *. d folder.

#obdiag rca list

Command	Description
obdiag rca run --scene=disconnection	root cause analysis of disconnection
obdiag rca run --scene=lock_conflict	root cause analysis of lock conflict
obdiag rca run --scene=ddl_disk_full	Insufficient disk space reported during DDL process
obdiag rca run --scene=major_hold	root cause analysis of major hold

Self-diagnosis Practice with obdiag Tool

One-click Root Cause Analysis

Example (disconnection scenario):

```
obdiag rca run --scene=disconnection
```

Result:

```
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
|                                                                record                                                                                     |
+------+----------------------------------------------------------------------------------------------------------------------------------------------------+
| step| info                                                                                                                                                 |
+-----+----------------------------------------------------------------------------------------------------------------------------------------------------+
|  1   | node:xxx.xxx.xxx obproxy_diagnosis_log:[2024-01-18 17:48:37.667014] [23173][Y0-00007FAA5183E710] |
|      | [CONNECTION](trace_type=“CLIENT_VC_TRACE”, connection_diagnosis={cs_id:1065, ss_id:4559,                                                             |
|      | proxy_session_id:837192278409543969, server_session_id:3221810838,                                                                                  |
|      | client_addr:“xxx.xxx.xxx.xxx:xxxx”, server_addr:“xxx.xxx.xxx.xxx:2883”, cluster_name:“obcluster”,                                                    |
|      | tenant_name:"sys", user_name:"root", error_code:-10010, error_msg:"An unexpected connection event |
|      | received from client while obproxy reading request", request_cmd:"COM_SLEEP", sql_cmd:"COM_END", |
|      | req_total_time(us):5315316}{vc_event:“VC_EVENT_EOS”, user_sql:“”})                                                                                   |
|  2   | cs_id:1065, server_session_id:3221810838                                                                                                             |
|  3   | trace_type:CLIENT_VC_TRACE                                                                                                                           |
|  4   | error_code:-10010                                                                                                                                    |
+-----+----------------------------------------------------------------------------------------------------------------------------------------------------+
```

The suggest: Need client cooperation for diagnosis

Self-diagnosis Practice with obdiag Tool

FROM INTRODUCTION TO PRACTICE

General Diagnostic Information Collection

General Information Collection:

```
obdiag gather <gather type> [options]
```

“gather type” includes the following	
log	Collect logs of the OceanBase cluster with one click
sysstat	Collect OceanBase cluster host information with one click
clog	Collect the clog logs of the OceanBase cluster with one click
slog	Collect slog logs of the OceanBase cluster with one click
plan_monitor	Collect the execution details of the SQL with the specified trace_id in the OceanBase cluster with one click
stack	Collect the stack trace information of the OceanBase cluster with one click
perf	Collect perf information of the OceanBase cluster with one click
obproxy_log	Collect the logs of the ODP that the OceanBase cluster depends on with one click
all	Collect diagnostic information of the OceanBase cluster with one click

```
#obdiag gather all
gather_all start ...
Download 192.168.1.1:/tmp/sysstat_192.168.1.1_20240426161848.zip
Downloading [====================] 100.0% [1.26 MB ]

Gather Sysstat Summary:
+-----------+-----------+----------+-------+-----------------------------------------------------------------------------------------------------------+
| Node      | Status    | Size     | Time| PackPath                                                                                                  |
+===========+===========+========+=====+===========================================================================================================+
| 192.168.1.1 | Completed | 1.260M | 4 s   | ./gather_pack_20240426161848/sysstat_192.168.1.1_20240426161848.zip |
+-----------+-----------+----------+-------+-----------------------------------------------------------------------------------------------------------+
Download 192.168.1.1:/tmp/obstack2_192.168.1.1_20240426161853.zip
Downloading [====================] 100.0% [12.05 KB ]

Gather Ob stack Summary:
+-----------+-----------+---------+---------+-----------------------------------------------------------------------------------------------------------+
| Node      | Status    | Size    | Time    | PackPath                                                                                                  |
+===========+===========+=========+=========+===========================================================================================================+
| 192.168.1.1 | Completed | 12.055K | 6 s     | ./gather_pack_20240426161848/obstack2_192.168.1.1_20240426161853.zip |
+-----------+-----------+---------+---------+-----------------------------------------------------------------------------------------------------------+
Download 192.168.1.1:/tmp/perf_192.168.1.1_20240426161859.zip
Downloading [====================] 100.0% [6.82 KB ]

Gather Perf Summary:
+-----------+-----------+---------+---------+-----------------------------------------------------------------------------------------------------------+
| Node      | Status    | Size    | Time    | PackPath                                                                                                  |
+===========+===========+=========+=========+===========================================================================================================+
| 192.168.1.1 | Completed | 6.818K  | 3 s     | ./gather_pack_20240426161848/perf_192.168.1.1_20240426161859.zip |
+-----------+-----------+---------+---------+-----------------------------------------------------------------------------------------------------------+
gather log from_time: 2024-04-26 15:49:03, to_time: 2024-04-26 16:20:03

Gather Ob Log Summary:
+-----------+-----------+----------+---------+-----------------------------------------------------------------------------------------------------------+
| Node      | Status    | Size     | Time    | PackPath                                                                                                  |
+===========+===========+==========+=========+===========================================================================================================+
| 192.168.1.1 | Completed | 35.134M | 13 s    | ./gather_pack_20240426161848/ob_log_192.168.1.1_20240426154903_20240426162003.zip |
+-----------+-----------+----------+---------+-----------------------------------------------------------------------------------------------------------+

If you want to view detailed obdiag logs, please run: obdiag display-trace 9b2c7a2e-03a5-11ef-9297-16c3aeac6f4f
```

Self-diagnosis Practice with obdiag Tool

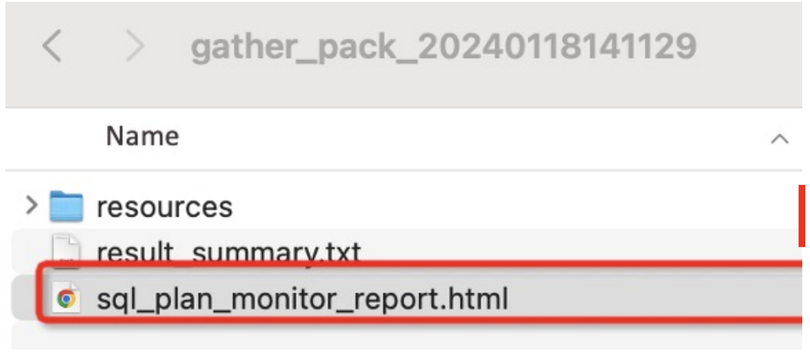
General Diagnostic Information Collection

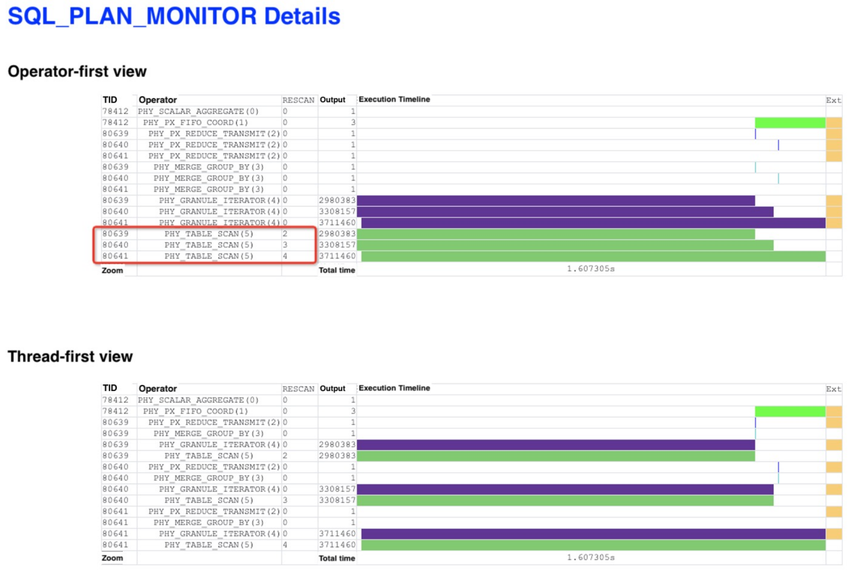
FROM INTRODUCTION TO PRACTICE

Stack trace information collection (the obstack2 tool is called, and the obdiag package comes with the obstack2 tool):
`obdiag gather stack`

```
#obdiag gather stack
gather_obstack start ...
Download 192.168.1.1:/tmp/obstack2_192.168.1.1_20240426163922.zip
Downloading [====================] 100.0% [12.06 KB ]
Gather Ob stack Summary:
+----------------+---------------+-----------+-------+-----------------------------------------------------------------------------------------------------------+
| Node           | Status        | Size      | Time | PackPath                                                                                                   |
+================+===============+===========+=======+===========================================================================================================+
| 192.168.1.1 | Completed | 12.056K | 6 s    | ./gather_pack_20240426163922/obstack2_192.168.1.1_20240426163922.zip |
+----------------+---------------+-----------+-------+-----------------------------------------------------------------------------------------------------------+
```

plan_monitor information collection:
`obdiag gather plan_monitor`





Self-diagnosis Practice with obdiag Tool

Scenario-based Information Collection

#obdiag gather scene list

Command	Information
obdiag gather scene run --scene=other.application_error	[application error]
obdiag gather scene run --scene=obproxy.restart	[obproxy restart]
obdiag gather scene run --scene=observer.backup	[backup problem]
obdiag gather scene run --scene=observer.backup_clean	[backup clean]
obdiag gather scene run --scene=observer.base	[cluster base info]
obdiag gather scene run --scene=observer.clog_disk_full	[clog disk full]
obdiag gather scene run --scene=observer.cluster_down	[cluster down]
obdiag gather scene run --scene=observer.compaction	[compaction]
obdiag gather scene run --scene=observer.cpu_high	[High CPU]
obdiag gather scene run --scene=observer.delay_of_primary_and_backup	[delay of primary and backup]
obdiag gather scene run --scene=observer.io	[io problem]
obdiag gather scene run --scene=observer.log_archive	[log archive]
obdiag gather scene run --scene=observer.long_transaction	[long transaction]
obdiag gather scene run --scene=observer.memory	[memory problem]
obdiag gather scene run --scene=observer.perf_sql --env "{db_connect='-h127.0.0.1 -P2881 -utest@test -p****** -Dtest', trace_id='Yxx'}"	[SQL performance problem]
obdiag gather scene run --scene=observer.px_collect_log --env "{trace_id='Yxx', estimated_time='2024-04-26 16:43:16'}"	[Collect error source node logs for SQL PX]
obdiag gather scene run --scene=observer.recovery	[recovery]
obdiag gather scene run --scene=observer.restart	[restart]
obdiag gather scene run --scene=observer.rootservice_switch	[rootservice switch]
obdiag gather scene run --scene=observer.sql_err --env "{db_connect='-h127.0.0.1 -P2881 -utest@test -p****** -Dtest', trace_id='Yxx'}"	[SQL execution error]
obdiag gather scene run --scene=observer.suspend_transaction	[suspend transaction]
obdiag gather scene run --scene=observer.unit_data_imbalance	[unit data imbalance]
obdiag gather scene run --scene=observer.unknown	[unknown problem]

Scenario-based information collection package list:
obdiag gather scene list

Scenario-based information collection and execution:
obdiag gather scene run --scene={SceneName}

Agenda

- Introduction to Agile Diagnostic Tools
- Use the Obdiag Tool for Self-diagnosis
- Add Obdiag Diagnostic Scenario by Yourself
- Agile Diagnostic Tools Summary and Outlook

Self-diagnosis Practice with obdiag Tool

Hot Update Scenario

No need to upgrade obdiag, you can quickly update the scene through obdiag update

```
#obdiag update -h
Usage: obdiag update [options]

Options:
  --file=FILE    obdiag update cheat file path. Please note that you need to ensure the reliability of the files on your own.
  --force        You can force online upgrades by adding --force in the command
  -h, --help     Show help and exit.
  -v, --verbose  Activate verbose output.


# Online network update, get the latest scene from the official website
obdiag update

# Offline update, update through offline scene package
obdiag update --file data.tar

#ls
backup_conf  check  check.d  config.yml  data.tar  data_version.yaml  example  gather  gather.d  log  rca  rca.d  remote_version.yaml
```

New scene data after obdiag update

Original secne data after obdiag update

Self-diagnosis Practice with obdiag Tool

FROM INTRODUCTION TO PRACTICE

Increased Inspection Scenarios

Inspection scene directory:
~/.obdiag/check

```
#tree
.
├── obproxy_check_package.yaml
├── observer_check_package.yaml
└── tasks
    ├── obproxy
    │   └── version
    │       └── bad_version.yaml
    └── observer
        ├── cluster
        │   ├── core_file_find.yaml
        │   ├── data_path_settings.yaml
        │   ├──...
        ├── cpu
        │   └── oversold.yaml
        ├── disk
        │   ├── clog_abnormal_file.yaml
        │   ├── ...
        ├── err_code
        │   ├── find_err_4000.yaml
        │   ├── ...
        ├── sysbench
        │   ├── sysbench_free_test_cpu_count.yaml
        │   ├── sysbench_free_test_memory_limit.yaml
        │   ├── ...
        ├── system
        │   ├── aio.yaml
        │   ├── ...
        └── ...
```

```
$ cat ~/.obdiag/check/tasks/observer/disk/disk_iops.yaml
info: "Check whether the disk iops."
task:
  - version: "[4.0.0.0,*]"
    steps:
    - type: sql
      sql: "SELECT GROUP_CONCAT(DISTINCT CONCAT(SVR_IP, ':', SVR_PORT) SEPARATOR ', ') AS
unique_server_endpoints
FROM oceanbase.GV$OB_IO_BENCHMARK
WHERE size=16384 AND IOPS<1024;"
      result:
        set_value: over_server
        verify: '[ -z "${over_server}" ]'
        err_msg: "These observer 16K IOPS are below 1024, please migrate as soon as possible. #{over_server}"
```

```
$ cat ~/.obdiag/check/tasks/observer/system/aio.yaml
info: 'To detect aio, refer to: https://www.oceanbase.com/docs/enterprise-oceanbase-ocp-cn-1000000000125643'
task:
  - steps:
      - type: ssh
        ssh: "ps -ef | grep observer | grep -v grep | wc -l"
        result:
          set_value: observer_nu
      - type: get_system_parameter
        parameter: fs.aio-max-nr
        result:
          set_value: aio_max_nr
          report_type: warning
          verify: "[ ${aio_max_nr} -ge 1048576 ]"
          err_msg: 'fs.aio-max-nr : #{aio_max_nr} is a non recommended value, recommended value  need  >1048576'
```

Self-diagnosis Practice with obdiag Tool

Collection Scene Added

FROM INTRODUCTION TO PRACTICE

Collection scene directory:
~/.obdiag/gather/task

```
.
├── obproxy
│   └── restart.yaml
├── observer
│   ├── backup_clean.yaml
│   ├── backup.yaml
│   ├── clog_disk_full.yaml
│   ├── cluster_down.yaml
│   ├── compaction.yaml
│   ├── delay_of_primary_and_backup.yaml
│   ├── io.yaml
│   ├── log_archive.yaml
│   ├── long_transaction.yaml
│   ├── memory.yaml
│   ├── recovery.yaml
│   ├── restart.yaml
│   ├── rootservice_switch.yaml
│   ├── suspend_transaction.yaml
│   ├── unit_data_imbalance.yaml
│   └── unknown.yaml
└── other
    └── application_error.yaml
```

Scenario Example: `obdiag gather scene run --scene=observer.memory`

```yaml
info_en: "[memory problem]"
command: obdiag gather scene run --scene=observer.memory
task:
- version: "[2.0.0.0, 4.0.0.0]"
steps:
- type: sql
sql: "SELECT * FROM oceanbase.v$ob_cluster"
global: true
- type: sql # View the 20 modules that use the most memory
sql: "select `CONTEXT`, ROUND(USED / 1024 / 1024 / 1024, 2) as USED_GB from oceanbase.gv$memory group by `CONTEXT`
      ORDER BY USED DESC limit 20;"
global: true
- type: sql # Check the memory status
sql: "select /*+ READ_CONSISTENCY(WEAK), query_timeout (100000000) */ TENANT_ID, round(ACTIVE / 1024 / 1024 / 1024, 2)
   ACTIVE_GB, round(TOTAL / 1024 / 1024 / 1024, 2) TOTAL_GB, round(FREEZE_TRIGGER / 1024 / 1024 / 1024, 2)
   FREEZE_TRIGGER_GB, round(TOTAL / FREEZE_TRIGGER * 100, 2) percent_trigger, round(MEM_LIMIT / 1024 / 1024 / 1024, 2)
   MEM_LIMIT_GB from oceanbase.gv$memstore where tenant_id > 1000 or TENANT_ID = 1 order by tenant_id, TOTAL_GB desc;"
global: true
- type: log
grep: ""
global: false
- version: "[4.0.0.0, *]"
steps:
- type: sql
sql: "SELECT * FROM oceanbase.DBA_OB_ZONES;"
global: true
- type: sql
sql: "select * from oceanbase.GV$OB_MEMSTORE limit 20"
global: true
- type: ssh # You can see the tenant's specifications, threads, queues, and request statistics.
           # This log is printed every 30 seconds for each tenant
ssh: "grep 'dump tenant info.tenant=' ${observer_data_dir}/log/observer.log | sed 's/,/,/g'"
global: false
- type: log
grep: ""
global: false
```

Self-diagnosis Practice with obdiag Tool

FROM INTRODUCTION TO PRACTICE

Increased Root Cause Analysis Scenarios

Root Cause Analysis Scenarios:
~/.obdiag/rca

```
#tree
.
├── ddl_disk_full_scene.py
├── disconnection_scene.py
├── lock_conflict_scene.py
└── major_hold_scene.py
```

Increased root cause analysis scenarios,
vi ~/.obdiag/rca/demo_scene.py

After adding root cause analysis scenarios,
obdiag rca run --scene=demo_scene.py

After adding root cause analysis scenarios,
obdiag rca list

```python
"""
@file: demo.py # Change it to your actual information
@desc: # Change it to your actual information
"""
# Since the framework uses a loader to import the library
class DemoScene(RcaScene):
    def __init__(self):
    super().__init__()

    # Initialize the root cause analysis class for parameter setting
    def init(self, context):

    # Perform root cause analysis
    def execute(self):
    # Get the root cause analysis results (including operation and
maintenance suggestions) and return them in RCA_ResultRecord format
    def get_result(self):
    # Set the return scenario usage instructions, required parameters, etc.
for scenario analysis
    def get_scene_info(self):
    # Class initialization, used by the framework to load this class
demo_scene = DemoScene()
```

obdiag rca list

Command	Info
obdiag rca run --scene=disconnection	root cause analysis of disconnection
obdiag rca run --scene=lock_conflict	root cause analysis of lock conflict
obdiag rca run --scene=ddl_disk_full	Insufficient disk space reported during DDL process
obdiag rca run --scene=major_hold	root cause analysis of major hold
obdiag rca run --scene=demo	root cause analysis of demo

Agenda

- Introduction to Agile Diagnostic Tools
- Use the Obdiag Tool for Self-diagnosis
- Add Obdiag Diagnostic Scenario by Yourself
- Agile Diagnostic Tools Summary and Outlook

Agile Diagnostic Tools Summary and Outlook

FROM INTRODUCTION TO PRACTICE

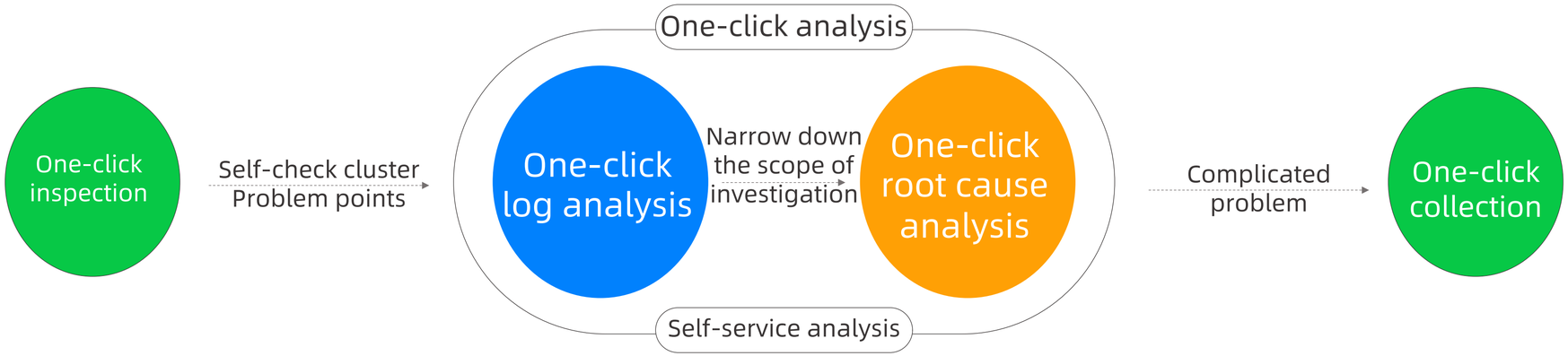
Summary

OceanBase Agile Diagnostic Tool:

obdiag

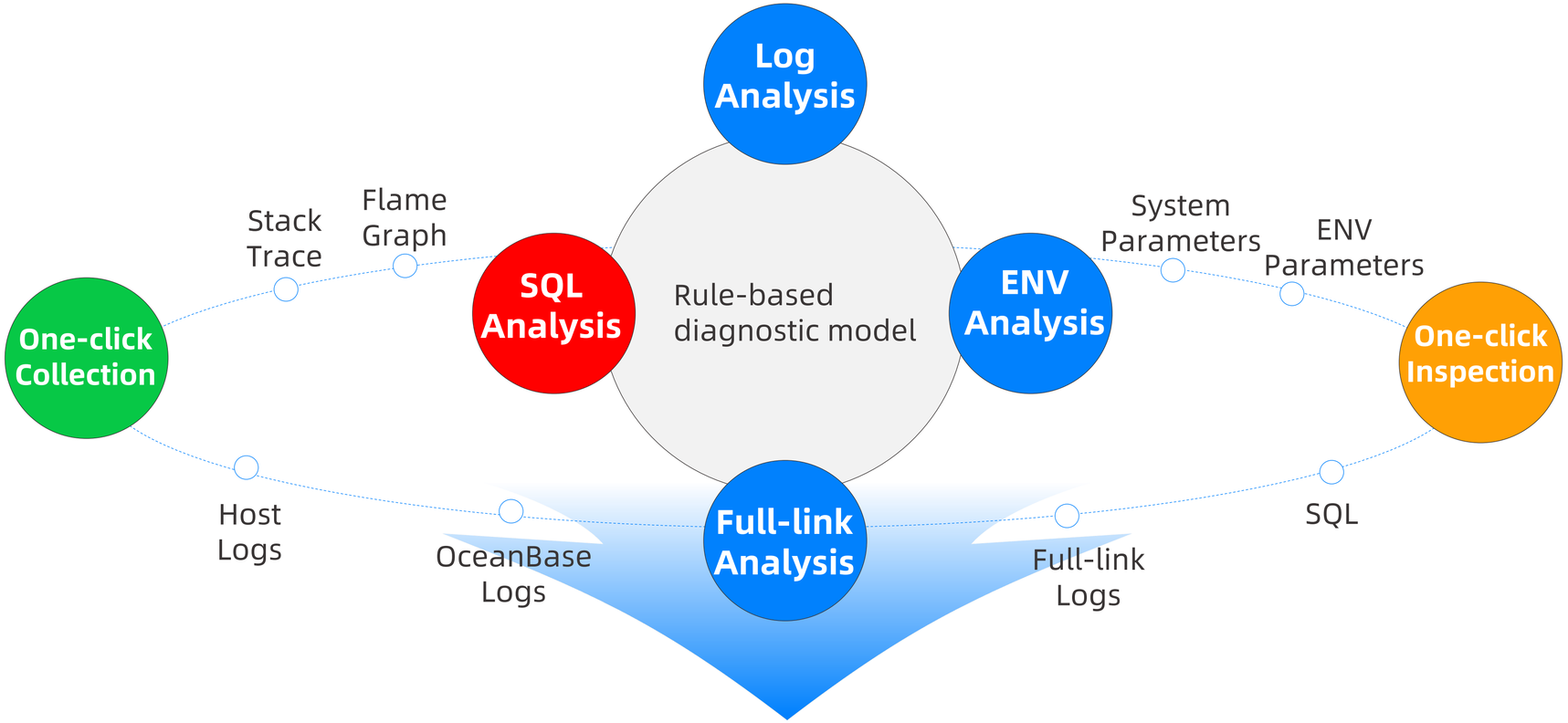
OB customized CLI diagnostic tool

Three tips for self-service troubleshooting of OceanBase database failure scenarios:



Agile Diagnostic Tools Summary and Outlook

obdiag Future Plans



SQL diagnoser capabilities are implemented in obdiag to complement obdiag's SQL diagnostic capabilities

Rely on one-click inspection to discover problems, use one-click collection to obtain diagnostic snapshots, and focus on rule-based analysis models to quickly diagnose OceanBase databases

FROM INTRODUCTION TO PRACTICE

Agile Diagnostic Tools Summary and Outlook

obdiag Future Plans

1.0.0

- One-click diagnostic information collection

1.3.0

- One-click log analysis

1.4.0

- One-click inspection

1.5.0

- One-click full-link diagnosis
- Sysbench stress test inspection
- Docker deployment cluster diagnosis

1.6.0

- Scenario-based diagnostic information collection
- Scenario-based root cause analysis

2.0.0

- Context transformation, enhanced scene expansion capability
- Root cause analysis phase II transformation

Published

Planning

2.1.0

- ash report collection
- Root cause analysis scenario expansion

2.2.0

- Root cause analysis scenario expansion
- tabledump collection

2.3.0

- Support SQL diagnosis
- Root cause analysis scenario expansion

2.4.0

- Adapt ob log compression transformation
- Phase 1 of service-oriented transformation
- Added adaptation for 2 kernel tools

2.5.0

- Add one-click diagnosis for OMS

3.0.0

- SQL Diagnosis Phase II, supports root cause analysis of SQL issues
- AI exploration

Oceanbase Open-Source Code Content

Item	Link
OceanBase	https://github.com/oceanbase/oceanbase
obproxy（ODP）	https://github.com/oceanbase/obproxy
obclient	https://github.com/oceanbase/obclient
OBD	https://github.com/oceanbase/obdeploy
ODC	https://github.com/oceanbase/odc
obdiag	https://github.com/oceanbase/obdiag
CDC Components	oblogproxy、oblogclient、oblogmsg
C Language Driver	ob-connector-odbc

For details, please see: https://github.com/orgs/oceanbase/repositories

OCEANBASE

FROM INTRODUCTION TO PRACTICE

Thank You!

OceanBase Official website:
https://oceanbase.github.io/

GitHub Discussions:
https://github.com/oceanbase/oceanbase/discussions