product-introduction

YHC (Yashan Health Check) | Yashan Deep Health Check Tool

YHC (Yashan Deep Health Check Tool) is a comprehensive inspection tool designed for YashanDB. It aims to provide professional database inspection reports and recommendations, offering flexible and complete solutions for customer data protection.

Target Users

- Yashan DBA

Product Positioning

- Lightweight standalone tool
- Out-of-the-box usage

Recommended Scenarios

- Performance monitoring and optimization
- Troubleshooting and problem diagnosis
- Security audit and compliance check
- Pre-upgrade and pre-migration database check
- Daily database inspection
- Any scenario requiring quick information retrieval

Core Features

Server Information Check

- Server basic information (operating system, hardware configuration, firewall, etc.)
- Server load status (network traffic, CPU usage, I/O load, memory, disk capacity, etc.)
- Server system logs
- ...

Database Information Check

- Database basic information (version, instance info, primary/standby info, etc.)
- Database file check (data files, control files, backup files, etc.)
- Database object check (tablespaces, tables, indexes, constraints, sequences, etc.)
- Database load check (sessions, transactions, wait events, lock waits, buffer pool hit rate, etc.)
- Database security check (password strength, user privileges, login configuration, default tablespace, etc.)
- Database log analysis (alert.log, run.log, redo logs, etc.)
- ...

Check Result Alerts and Recommendations

- Support for custom metric alert expressions
- Flexible configurable thresholds, alert levels, and alert recommendations

Rich and Extensible Check Metrics

- Provides rich default metrics
- Supports custom check metrics (bash, sql)

Flexible Configurable Check Strategies

- Support for custom time periods with absolute flexibility in time period selection
- Support for custom path data checks, batch selection of directories or files
- ...

Rich and Complete Data Management

- Support for custom check data storage path
- Multiple check data display formats (html, docx)
- ...

specifications

Product Form

A standalone command-line tool

Platform Support

- Linux (ubuntu/centos/kylin)

Hardware Architecture

- x86
- arm

Deployment Method

- No deployment required, out-of-the-box usage

Product Specifications

1. Only supports local information check
2. Only supports information check for YashanDB SE

Parameter Specifications

The following constraints apply to the `check` and `after-install` subcommands:

- The -r parameter has higher priority than -s/-e
- -r maximum is 30d, minimum is 1m, can be modified through configuration file ({yhc_home}/config/yhc.toml)
- -s/-e format is yyyy-MM-dd-ss-mm, maximum interval is 7d, minimum is 1m, can be modified through configuration file ({yhc_home}/config/yhc.toml)

Feature Specifications

- The tool will attempt to check the server's historical load data for the specified time period (default 24h). This depends on sar (System Activity Reporter), a common system performance monitoring tool. We recommend pre-installing this tool; otherwise, historical load data cannot be checked. Valid data can only be tracked after the sar tool starts running, and by default only saves data within 30 days.

quick-start

Tool Acquisition

1. Log in to the server where the database is located and access the URL to get the tool package

```
http://192.168.19.121:9988/product/ycm/release/yhc/v0.1.6/
```

2. Extract to the corresponding directory (example extracts to the current directory by default):

```
# Using x86_64 architecture package as an example
tar -zxvf yashan-health-check-v0.1.6-linux-x86_64.tar.gz
```

3. Enter the `yashan-health-check-v0.1.6` folder and execute `./yhcctl -v` to verify

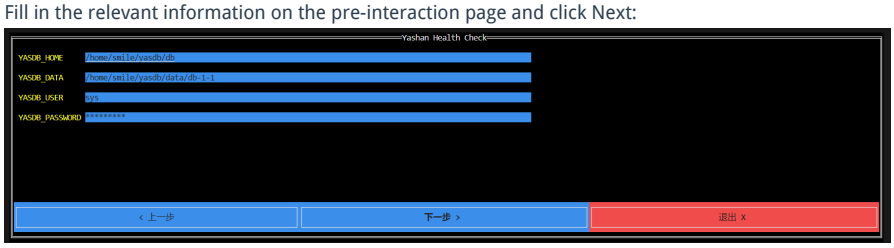
```
-bash-4.2$ ./yhcctl -v
v0.1.6
```

Start Check

It is strongly recommended to run this tool with the user who owns the database process and grant sudo privileges to that user, or run the tool as root user.

Execute `./yhcctl check` (or `./yhcctl after-install`) to start a check directly. When no parameters are specified, the tool will check information within the last 24 hours by default (for specific check items, please refer to Reference Manual).

1. Fill in the relevant information on the pre-interaction page and click Next:



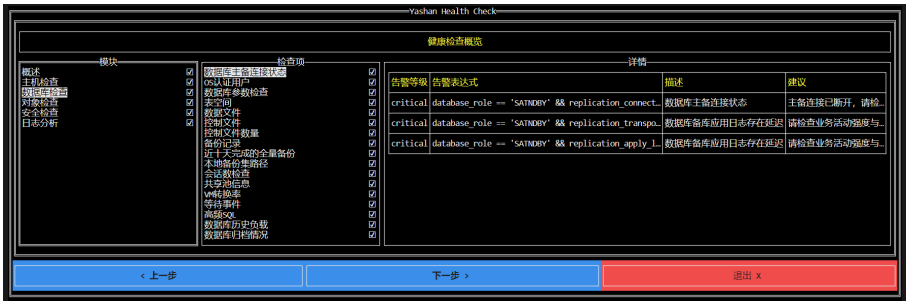

```
- YASDB_HOME: Default fills with the current environment variable YASDB_HOME value. If not available, it will try to match
the YASDB_HOME path of the existing yashandb process. Users can modify and confirm.
- YASDB_DATA: Default fills with the current environment variable YASDB_DATA value. If not available, it will try to match
the YASDB_DATA path of the existing yashandb process. Users can modify and confirm.
- YASDB_USER: Users need to provide a username that can access the current database, with at least CREATE SESSION privilege.
DBA user or SYS user is recommended. Users can modify and confirm.
- YASDB_PASSWORD: Users need to provide the valid password for the YASDB_USER. Users can modify and confirm.
If the operating system user belongs to the YASDBA user group, health check can be performed without entering database
username and password.
```

2. The tool will verify whether the expected check data can be correctly checked based on the information provided by the user, and display reasons for items that cannot be checked, as shown below:

> Common reasons include:

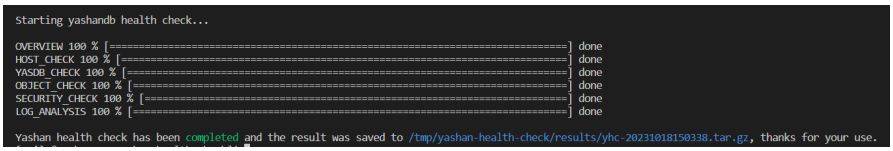
> Common reasons include:
> - 1. Current check item requires root privileges: because executing certain commands or operating certain files requires root privileges.
> - 2. DBA privileges required: because the corresponding check item requires at least DBA privileges for the database user.
> - 3. System audit needs to be enabled: because when audit is disabled (parameter: UNIFIED_AUDITING), the system will not perform any auditing.
> - 4. User xxx does not have xxx access permission: because the current user does not have access permission to the operating system file. Without -r-x permission, this information cannot be checked. We recommend using a user with expected permissions to run the tool for more comprehensive information checking. Recommended: database owner user with sudo privileges.

3. Click Next to continue checking, or click Exit to change the check user according to the reasons. As shown below (example shows clicking Next to continue):



> Before starting the check, all check items that can be checked will be listed, including module, check item, check item alerts, etc. For specific check items, please refer to Reference Manual.
> All check items are selected by default. If there are items you don't want to check, you can deselect them.

4. Click Next to start checking, observe the specific progress, and wait for the check to complete, as shown below:



> The check progress will be dynamically updated. After completion, the result storage path will be displayed. This result is stored at `/tmp/yashan-health-check/results/yhc-20231018150338.tar.gz`. By default, check results will be packaged and stored in the results directory under the tool execution directory (relative path). Users can also use the -o parameter to customize the path.

Non-Interactive Mode

Execute `./yhcctl check -d` to enable non-interactive mode, which will confirm all pending items by default.

In non-interactive mode, you need to specify user and password through parameters, and also specify `YASDB_DATA` and `YASDB_HOME`. If not specified, environment variable values will be used by default.

For example, with `YASDB_DATA` and `YASDB_HOME` environment variables set, you can execute `./yhcctl check -d -u xxx -p xxx` to start the health check.

Without environment variables set, you can execute `./yhcctl check -d -u xxx -p xxx --yasdb-home xxx --yasdb-data xxx` to start the health check.

For detailed parameters, run `./yhcctl check -h`.

Language Support

YHC tool supports both Chinese and English from version v0.1.7. You can specify the language in two ways:

Method 1: Command Line Parameter (Highest Priority)

Use the `-l` or `--language` parameter to specify the language:

```
# Use Chinese
./yhcctl check -l zh

# Use English
./yhcctl check -l en
```

Method 2: Configuration File

Set the `language` field in the configuration file `{yhc_home}/config/yhc.toml`:

```
# Use Chinese (default)
language = "zh"

# Use English
language = "en"
```

Priority

- Command line parameter > Configuration file > Default value (Chinese)
- If both command line parameter and configuration file are specified, the command line parameter overrides the configuration file setting
- If neither is specified, Chinese is used by default

Language Setting Scope

The language setting affects:

- CLI command output and prompts
- HTML report interface
- Word report content

In English mode, all CLI output (including progress bars, prompts, error messages, etc.) will be displayed in English:

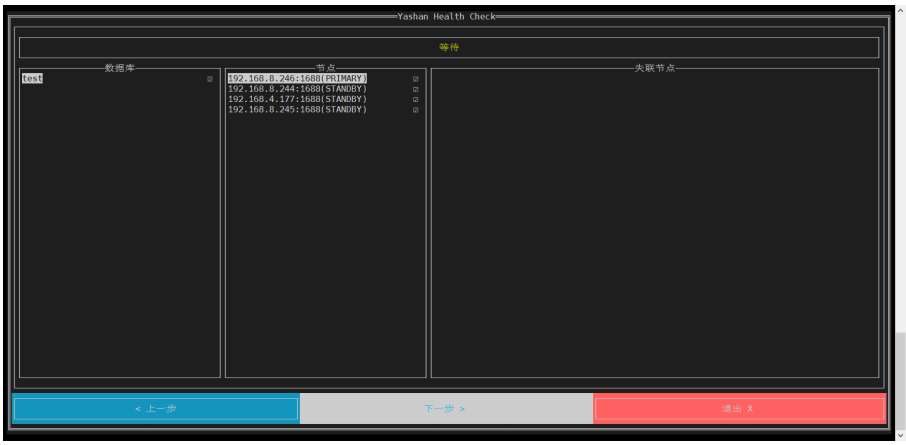
```
Starting yashandb health check...

OVERVIEW 100 % [==================] done
HOST CHECK 100 % [==================] done
DATABASE CHECK 100 % [==================] done

Packing check results, please wait for a moment...

The result was saved to /path/to/yhc-20231125163822.tar.gz, thanks for your use.
```

Multi-Node Mode

Execute `./yhcctl check -m` to enable multi-node mode. Multi-node mode will obtain information about all nodes in the cluster to be checked usin g the yasboot tool under the current YASDB_HOME and verify node connectivity.

You can also specify nodes to check through the configuration file. For details, see Configuration File.

In interactive mode, a node information interface will be displayed for users to select nodes to check, as shown below:



In non-interactive mode, all connectable nodes will be checked by default.

Viewing Information

1. Extract the check information package, which will generate a directory with the same name:

```
# Extract file
-bash-4.2$ tar zxvf yhc-20231018150338.tar.gz
yhc-20231018150338/
yhc-20231018150338/data/
yhc-20231018150338/data/data-20231018150338.json
yhc-20231018150338/data/report-20231018150338.json
yhc-20231018150338/data/failed-20231018150338.json
yhc-20231018150338/report-20231018150338.html
yhc-20231018150338/report-20231018150338.docx

# View current directory
-bash-4.2$ ll
total 1196
drwxr-xr-x 3 smile smile      86 Oct 18 15:04 yhc-20231018150338
-rw-r--r-- 1 smile smile 1221824 Oct 18 15:04 yhc-20231018150338.tar.gz

# Enter yhc-20231018150338 directory
-bash-4.2$ cd yhc-20231018150338/
-bash-4.2$ ll
total 3436
drwxr-xr-x 2 smile smile     106 Oct 18 15:04 data
-rw-r--r-- 1 smile smile  228794 Oct 18 15:04 report-20231018150338.docx
-rw-r--r-- 1 smile smile 3288159 Oct 18 15:04 report-20231018150338.html
```

- data: Source data folder for this check, containing raw data in `json` format, which serves as the data source for generating check reports
- report-{generation time}.html: Deep check report in `html` format
- report-{generation time}.docx: Deep check report in `docx` format

> For more information, please refer to Report Interpretation

best-practices

This section provides minimal unit guidance for tool usage scenarios. Users can integrate scenario units according to actual needs to form best practices.

Parameter Explanation

YHC currently supports two subcommands: `check` and `after-install`.

```
Usage: yhcctl <command>

Yhcctl is used to manage the yashan health check.

Flags:
  -h, --help                          Show context-sensitive help.
  -v, --version                       Show version.
  -c, --config="./config/yhc.toml"    Configuration file.

Commands:
  check            The check command is used to yashan health check.

  after-install    The after-install command is used to verify the installation of Yashandb after it has been installed.

Run "yhcctl <command> --help" for more information on a command.
```

The check command is generally used for daily database health checks. Below is parameter explanation for the check command.

```
-bash-4.2$ ./yhcctl check -h
Usage: yhcctl check

The check command is used to yashan health check.

Flags:
  -h, --help                          Show context-sensitive help.
  -v, --version                       Show version.
  -c, --config="./config/yhc.toml"    Configuration file.

  -r, --range=STRING                  The time range of the check, such as '1M', '1d', '1h', '1m'. If <range> is given,
<start> and <end> will be discard.
  -s, --start=STRING                  The start datetime of the check, such as 'yyyy-MM-dd', 'yyyy-MM-dd-hh', 'yyyy-MM-dd-hh-
mm'
  -e, --end=STRING                    The end timestamp of the check, such as 'yyyy-MM-dd', 'yyyy-MM-dd-hh', 'yyyy-MM-dd-hh-
mm', default value is current datetime.
  -o, --output=STRING                 The output dir of the check.
  -d, --disable-interaction           Disable interaction.
  -m, --multiple-nodes                Check multiple nodes.
      --yasdb-home=STRING             Home path of YashanDB(env: YASDB_HOME).
      --yasdb-data=STRING             Data path of YashanDB(env: YASDB_DATA).
  -u, --user=STRING                   YashanDB user for checking.
  -p, --password=STRING               YashanDB user password for checking.
```

- -h: View help information for check subcommand
- -v: View current version of yhc tool
- -c: Specify configuration file used by tool, default is `{yhc_home}/config/yhc.toml`
- -r: Specify time period, default checks data within 24h before current time. This option takes global effect and has higher priority than start/end parameters. Constraint: max 30d, min 1m, can be modified via config file ({yhc_home}/config/yhc.toml)

- -s: Specify check data start time, default is 24h before current time. This option takes global effect and has lower priority than -r parameter. Constraint: format `yyyy-MM-dd-ss-mm`, max interval with end 7d, min 1m, can be modified via config file ({yhc_home}/config/yhc.toml)
- -e: Specify check data end time, default is current time. This option takes global effect and has lower priority than -r parameter. Constraint: format `yyyy-MM-dd-ss-mm`, max interval with start 30d, min 1m, can be modified via config file ({yhc_home}/config/yhc.toml)
- -o: Specify path for storing checked data. This path requires at least write permission for current tool execution user
- -d: Disable interactive mode. In this mode, terminal interaction will not be enabled, instead using command line parameters for user password input
- -m: Check multiple nodes. By default, yhc tool checks single node. This parameter enables checking multiple nodes
- --yasdb-home: Specify yashandb home path. This parameter has higher priority than environment variable `YASDB_HOME` in yhc tool, typically used in non-interactive mode
- --yasdb-data: Specify yashandb data path. This parameter has higher priority than environment variable `YASDB_DATA` in yhc tool, typically used in non-interactive mode
- -u: Specify yashandb username, typically used in non-interactive mode
- -p: Specify yashandb user password, typically used in non-interactive mode

The after-install command is used to check database status after database deployment is complete.

```
-bash-4.2$ ./yhcctl after-install -h
Usage: yhcctl after-install

The after-install command is used to verify the installation of Yashandb after it has been installed.

Flags:
  -h, --help                          Show context-sensitive help.
  -v, --version                       Show version.
  -c, --config="./config/yhc.toml"    Configuration file.

  -r, --range=STRING                  The time range of the check, such as '1M', '1d', '1h', '1m'. If <range> is given,
<start> and <end> will be discard.
  -s, --start=STRING                  The start datetime of the check, such as 'yyyy-MM-dd', 'yyyy-MM-dd-hh', 'yyyy-MM-dd-hh-
mm'
  -e, --end=STRING                    The end timestamp of the check, such as 'yyyy-MM-dd', 'yyyy-MM-dd-hh', 'yyyy-MM-dd-hh-
mm', default value is current datetime.
  -o, --output=STRING                 The output dir of the check.
  -d, --disable-interaction           Disable interaction.
  -m, --multiple-nodes                Check multiple nodes.
      --yasdb-home=STRING             Home path of YashanDB(env: YASDB_HOME).
      --yasdb-data=STRING             Data path of YashanDB(env: YASDB_DATA).
  -u, --user=STRING                   YashanDB user for checking.
  -p, --password=STRING               YashanDB user password for checking.
```

- -h: View help information for after-install subcommand
- -v: View current version of yhc tool
- -c: Specify configuration file used by tool, default is `{yhc_home}/config/yhc.toml`
- -r: Specify time period, default checks data within 24h before current time. This option takes global effect and has higher priority than start/end parameters. Constraint: max 30d, min 1m, can be modified via config file ({yhc_home}/config/yhc.toml)
- -s: Specify check data start time, default is 24h before current time. This option takes global effect and has lower priority than -r parameter. Constraint: format `yyyy-MM-dd-ss-mm`, max interval with end 7d, min 1m, can be modified via config file ({yhc_home}/config/yhc.toml)
- -e: Specify check data end time, default is current time. This option takes global effect and has lower priority than -r parameter. Constraint: format `yyyy-MM-dd-ss-mm`, max interval with start 30d, min 1m, can be modified via config file ({yhc_home}/config/yhc.toml)
- -o: Specify path for storing checked data. This path requires at least write permission for current tool execution user
- -d: Disable interactive mode. In this mode, terminal interaction will not be enabled, instead using command line parameters for user password input
- -m: Check multiple nodes. By default, yhc tool checks single node. This parameter enables checking multiple nodes
- --yasdb-home: Specify yashandb home path. This parameter has higher priority than environment variable `YASDB_HOME` in yhc tool, typically used in non-interactive mode
- --yasdb-data: Specify yashandb data path. This parameter has higher priority than environment variable `YASDB_DATA` in yhc tool, typically used in non-interactive mode
- -u: Specify yashandb username, typically used in non-interactive mode
- -p: Specify yashandb user password, typically used in non-interactive mode

default-check

Simply execute `yhcctl check`

Default check means not specifying any parameters, all parameters use default values. For specific default values, see Parameter Explanation

For specific execution guidance for default check, see Quick Start

specify-time-range

Tool Acquisition

1. Log in to the server where the database is located and access the URL to get the tool package

```
http://192.168.19.121:9988/product/ycm/release/yhc/v0.1.6/
```

2. Extract to the corresponding directory (example extracts to the current directory by default):

```
# Using x86_64 architecture package as an example
tar -zxvf yashan-health-check-v0.1.6-linux-x86_64.tar.gz
```

3. Enter the `yashan-health-check-v0.1.6` folder and execute `./yhcctl -v` to verify

```
-bash-4.2$ ./yhcctl -v
v0.1.6
```

Start Check

> The following are two ways to specify time range. The -r option has higher priority than -s/-e, meaning if -r parameter is specified, -s/-e will not take effect for this execution.

Specify via -r Parameter

- Usage example: `-r 1d` means within one day, `-r 1h` means within one hour, `-r 1m` means within 1 minute
- -r maximum is 30d, minimum is 1m, can be modified through configuration file ({yhc_home}/config/yhc.toml)

> For example, execute `./yhcctl check -r 7h` to specify checking data information from 7 hours before current time.

Specify via -s/-e Parameters

- Usage example: `-s 2023-06-01 -e 2023-07-01` means checking data from June 1st to July 1st
- -s/-e format is yyyy-MM-dd-ss-mm, maximum interval is 7d, minimum is 1m, can be modified through configuration file ({yhc_home}/config/yhc.toml)
- -s and -e can be specified individually. Unspecified parameters use default values: -s defaults to 24h before current time, -e defaults to current time

> For example, execute `./yhcctl check -s 2023-06-01` to specify start time as June 1st 2023, while end time uses default current time. However, ensure time period length doesn't exceed maximum spec of 30d. Adjust corresponding configuration file ({yhc_home}/config/yhc.toml) if needed.

Viewing Information

Based on the report storage path shown after tool execution completes successfully, find the corresponding check result to view detailed information of this check data (for specific report interpretation, see Report Interpretation).

specify-output-path

Tool Acquisition

1. Log in to the server where the database is located and access the URL to get the tool package

```
http://192.168.19.121:9988/product/ycm/release/yhc/v0.1.6/
```

2. Extract to the corresponding directory (example extracts to the current directory by default):

```
# Using x86_64 architecture package as an example
tar -zxvf yashan-health-check-v0.1.6-linux-x86_64.tar.gz
```

3. Enter the `yashan-health-check-v0.1.6` folder and execute `./yhcctl -v` to verify

```
-bash-4.2$ ./yhcctl -v
v0.1.6
```

Start Check

Specify via -o Parameter

- -o: Supports specifying file storage path after check completion. If path doesn't exist, tool will create it automatically. If current user lacks permissi on, tool cannot continue running.

> For example, execute `./yhcctl check -o /tmp/results` to store this check's data package at `/tmp/results` path.

Use Default Path

- When not using -o parameter to specify file storage path after check completion, the output configuration in config file will be used as default stor age path, defaulting to `results` folder under `yashan-health-check` directory.

Viewing Information

Based on the report storage path shown after tool execution completes successfully, find the corresponding check result to view detailed information of this check data (for specific report interpretation, see Report Interpretation).

custom-check-items

Tool Acquisition

1. Log in to the server where the database is located and access the URL to get the tool package

```
http://192.168.19.121:9988/product/ycm/release/yhc/v0.1.6/
```

2. Extract to the corresponding directory (example extracts to the current directory by default):

```
# Using x86_64 architecture package as an example
tar -zxvf yashan-health-check-v0.1.6-linux-x86_64.tar.gz
```

3. Enter the `yashan-health-check-v0.1.6` folder and execute `./yhcctl -v` to verify

```
-bash-4.2$ ./yhcctl -v
v0.1.6
```

Start Configuration

Using Default Custom File

YHC provides default custom metric configuration file `{yhc_home}/config/custom_metric.toml`, where you can define directly.

YHC currently supports two types of custom metrics: `sql` and `bash`.

Custom SQL Metric

An example is shown below, defining the SQL statement to execute, column name aliases for display, and alert configuration:

```
[[metrics]]
  name = "custom_check_sql"
  name_alias = "Custom SQL Metric"
  module_name = "custom_check"
  enabled = true
  metric_type = "sql"
  sql = "select status as instance_status, version, startup_time from v$instance;"
  [metrics.column_alias]
    INSTANCE_STATUS = "Database Instance Status"
    STARTUP_TIME = "Database Instance Startup Time"
    VERSION = "Database Version"
  [metrics.item_names]
    INSTANCE_STATUS = "instance_status"
  [metrics.alert_rules]

    [[metrics.alert_rules.critical]]
      expression = "instance_status != 'OPEN'"
      description = "Instance status abnormal"
      suggestion = "Recommend checking instance status"
```

To learn more about alert expression usage, please refer to Alert Expression for guidance.

Custom Bash Metric

An example is shown below, executing `uname -a` command, with command result displayed as raw string.

> Parsing Bash output into table format and Bash metric alerts are not yet supported.

```
[[metrics]]
  name = "custom_check_bash"
```

```
  name_alias = "Custom BASH Metric"
  module_name = "custom_check"
  enabled = true
  command = "uname -a"
  metric_type = "bash"
```

Using Other Custom Files

You can configure custom metrics in other files, then add the configuration file path to the `metric_paths` field in `{yhc_home}/config/yhc.toml`.

> For example, if new metrics are defined in `/tmp/sql_metrics.toml` file, modify `metric_paths` field to ["./config/default_metric.toml", "./config/custom_metric.toml", "/tmp/sql_metrics.toml"]
> Also note file access permissions, ensure at least read permission for configuration files

Configuring Report Module

After configuring metrics, custom metrics will be checked during execution. To display metrics in report, report module also needs to be configured.

Open report module configuration file `{yhc_home}/config/report_module.toml`, find custom check items, and write new metric configuration into report module.

For example, defining two metrics `custom_check_bash` and `custom_check_sql`, configuration file is as follows:

```
[[modules]]
  name = "custom_check"
  name_alias = "Custom Check"

  [[modules.children]]
    name = "custom_check_bash"
    name_alias = "Custom BASH Metric"
    metric_names = ["bash_test"]

  [[modules.children]]
    name = "custom_check_sql"
    name_alias = "Custom SQL Metric"
    metric_names = ["sql_test"]
```

Verification

After configuration is complete, execute a check to verify whether custom metrics are configured correctly and displayed correctly in report.

reference

This section provides detailed explanations of the tool's built-in data check implementations, post-check data package interpretation, and tool configuration guidance, helping users better understand and analyze the checked data and control tool capabilities.

tool-structure

Tool Acquisition

1. Log in to the server where the database is located and access the URL to get the tool package

```
http://192.168.19.121:9988/product/ycm/release/yhc/v0.1.6/
```

2. Extract to the corresponding directory (example extracts to the current directory by default):

```
# Using x86_64 architecture package as an example
tar -zxvf yashan-health-check-v0.1.6-linux-x86_64.tar.gz
```

3. Enter the `yashan-health-check-v0.1.6` folder and execute `./yhcctl -v` to verify

```
-bash-4.2$ ./yhcctl -v
v0.1.6
```

Tool Structure

```
-bash-4.2$ cd yashan-health-check
-bash-4.2$ ll
drwxrwxr-x 2 smile smile  20 Oct 17 09:37 bin
drwxrwxr-x 2 smile smile 101 Oct 17 09:37 config
drwxrwxr-x 2 smile smile   6 Oct 17 09:37 docs
drwxrwxr-x 2 smile smile  27 Oct 17 09:37 html-template
drwxrwxr-x 2 smile smile  43 Oct 17 09:37 log
drwxrwxr-x 3 smile smile  65 Oct 18 15:08 results
drwxrwxr-x 3 smile smile  40 Oct 17 09:37 scripts
lrwxrwxrwx 1 smile smile  12 Oct 17 09:37 yhcctl -> ./bin/yhcctl
```

The corresponding content of each folder and file is as follows:

File/Folder	Description
bin	Executable binary file directory
config	Configuration file storage directory
docs	Tool documentation storage directory
html-template	Template files for generating HTML reports
log	Tool log storage directory
results	Default check result storage directory
scripts	Tool script storage directory
yhcctl	Tool entry point, symlink pointing to ./bin/yhcctl

overview

Host Overview

Check Item	Implementation	Notes
Host Information	Calls `gopsutil` package's `host.Info()`	Underlying calls OS basic commands
Disk Information	Calls `gopsutil` package's `disk.Partitions()` and `disk.Usage()`	Underlying calls OS basic commands
Disk Block Device	Executes `lsblk -P -b` command	
Memory Information	Calls `gopsutil` package's `mem.VirtualMemory()`	Underlying calls OS basic commands
Network Interface	Calls `gopsutil` package's `net.InterfacesI()`	Underlying calls OS basic commands
BIOS Information	Executes `dmidecode` command	Requires Root privileges

Database Overview

Check Item	Implementation	Notes

Database Information	Queries database views: `v$instance` , `v$database` , `v$parameter` , to get database basic information
	v$instance
	`select status as instance_status, version, startup_time from v$instance;`
	v$database
	`select database_name, status as database_status, log_mode, open_mode, database_role, protection_mode, protection_level, create_time from v$database;`
	v$parameter
	Listen address
	`select VALUE as LISTEN_ADDR from v$parameter where name = 'LISTEN_ADDR';`
	Deployment topology
	`select count(*) as node_num from v$parameter where value is not null and name like '%ARCHIVE_DEST%';`
		If database status is abnormal, some information may not be available

Database File Permissions	Calls golang built-in library `filepath` for database file permission check	Underlying calls OS basic commands
Database Deployment Topology	`select count(*) as node_num from v$parameter where value is not null and name like '%ARCHIVE_DEST%';`	
Database Archive Clean Threshold	`SELECT * FROM V$PARAMETER WHERE NAME = 'ARCH_CLEAN_UPPER_THRESHOLD' UNION SELECT * FROM V$PARAMETER WHERE NAME = 'ARCH_CLEAN_LOWER_THRESHOLD';`	

host-check

Host Load Check

Check Item	Implementation	Notes
CPU Usage	1. Historical load (depends on sar tool) Check if sar has custom path configuration `sar_dir`: `/etc/sysconfig/sysstat` If not, use OS default path (ubuntu: `/var/log/sysstat`; others: `/var/log/sa`) Execute query command: `sar -u -f {sar_dir}/{sa file for date} -s 03:41:57 -e 15:41:57` 2. Current load If sar exists, query current network load via sar, e.g., execute `sar -u 1 10` to query once per second, 10 times total If sar doesn't exist, use gopsutil's net.INCounters() to query current network load, once per second, 10 times total, then calculate approximate load	1. Cannot get historical load data when sar is not installed, only current load data 2. gopsutil uses custom algorithm with slight deviation, for reference only 3. Interval and count can be adjusted via config file
Disk Usage	1. Historical load (depends on sar tool) Check if sar has custom path configuration `sar_dir`: `/etc/sysconfig/sysstat` If not, use OS default path (ubuntu: `/var/log/sysstat`; others: `/var/log/sa`) Execute query command: `sar -d -f {sar_dir}/{sa file for date} -s 03:41:57 -e 15:41:57` 2. Current load If sar exists, query current disk load via sar, e.g., execute `sar -d 1 10` to query once per second, 10 times total If sar doesn't exist, use gopsutil to calculate approximate load	1. Cannot get historical load data when sar is not installed, only current load data 2. gopsutil uses custom algorithm with slight deviation, for reference only 3. Interval and count can be adjusted via config file
Memory Usage	1. Historical load (depends on sar tool) Check if sar has custom path configuration `sar_dir`: `/etc/sysconfig/sysstat` If not, use OS default path (ubuntu: `/var/log/sysstat`; others: `/var/log/sa`) Execute query command: `sar -r -f {sar_dir}/{sa file for date} -s 03:41:57 -e 15:41:57` 2. Current load If sar exists, query current memory load via sar, e.g., execute `sar -r 1 10` to query once per second, 10 times total If sar doesn't exist, use gopsutil to calculate approximate load	1. Cannot get historical load data when sar is not installed, only current load data 2. gopsutil uses custom algorithm with slight deviation, for reference only 3. Interval and count can be adjusted via config file
Network Usage	1. Historical load (depends on sar tool) Check if sar has custom path configuration `sar_dir`: `/etc/sysconfig/sysstat` If not, use OS default path (ubuntu: `/var/log/sysstat`; others: `/var/log/sa`) Execute query command: `sar -n DEV -f {sar_dir}/{sa file for date} -s 03:41:57 -e 15:41:57` 2. Current load If sar exists, query current network load via sar, e.g., execute `sar -n DEV 1 10` to query once per second, 10 times total If sar doesn't exist, use gopsutil's net.INCounters() to calculate approximate load	1. Cannot get historical load data when sar is not installed, only current load data 2. gopsutil uses custom algorithm with slight deviation, for reference only 3. Interval and count can be adjusted via config file
Firewall Status	1. Ubuntu system, execute `ufw status` command 2. Other systems, execute `systemctl is-active firewalld` command	
Port Open Status	Execute `iptables -L` command	

database-check

Primary/Standby Check

Check Item	Implementation	Notes
Database Primary/Standby Connection Status	Query `v$replication_status` view: `select connection, status, peer_role, peer_addr, transport_lag, apply_lag from v$replication_status;`	

Database Configuration Check

Check Item	Implementation	Notes
Database Parameter Check	Query `v$parameter` view: `select name, value from v$parameter where value is not null;`	
OS Authentication Users	1. OS authentication switch Check `ENABLE_LOCAL_OSAUTH` configuration in `${YASDB_DATA}/config/yasdb_net.ini` 2. Users in YASDBA group Parse `/etc/group` file to find users in target group	

Tablespace Check

Check Item	Implementation	Notes
Tablespace	Query `dba_tablespaces` view: `SELECT TABLESPACE_NAME, CONTENTS, STATUS, ALLOCATION_TYPE , TOTAL_BYTES - USER_BYTES AS USED_BYTES, TOTAL_BYTES, (TOTAL_BYTES - USER_BYTES) / TOTAL_BYTES * 100 AS USED_RATE FROM SYS.DBA_TABLESPACES;`	
Data Files	Query `dba_data_files` view: `select * from dba_data_files` Call `os.Stat()` to get data file permission information	

Control File Check

Check Item	Implementation	Notes
Control File	Query `v$controlfile` view: `select id, name, bytes/1024/1024 as MBytes from v$controlfile;`	
Control File Count	Query `v$controlfile` view: `select count(*) as total from v$controlfile;`	

Backup Check

Check Item	Implementation	Notes
Backup Records	Query `dba_backup_set` view: `select RECID# as RECID, START_TIME, TYPE, decode(COMPLETION_TIME > sysdate, FALSE, TRUE) as SUCCESS from dba_backup_set;`	

Check Item	Implementation	Notes
Full Backups Completed in Last 10 Days	Query `dba_backup_set` view: `select count(*) as TOTAL from dba_backup_set where date_add(COMPLETION_TIME , INTERVAL 10 DAY) >= sysdate AND type = 'FULL';`	
Local Backup Set Path	Query `dba_backup_set` view: `"select distinct(PATH) as PATH from dba_backup_set;` Call `os.Stat()` to check if backup file exists	

Load Check

Check Item	Implementation	Notes
Session Count	Query `v$session` view: `select * from v$session;`	
Shared Pool Info	Query `v$sgastat` view: `select NAME, BYTES from v$sgastat WHERE POOL='SHARE POOL';`	

Archive Log Check

Check Item	Implementation	Notes
Database Archive Status	Query `v$archived_log, V$PARAMETER, V$SYSTEM_PARAMETER` views: `WITH value_stats AS (SELECT SUM(BLOCK_SIZE * BLOCKS) AS total_blocks FROM V$ARCHIVED_LOG), arch_clean_upper_threshold AS (SELECT CASE WHEN VALUE LIKE '%T' THEN TRIM(TRAILING 'T' FROM VALUE) * 1024 * 1024 * 1024 * 1024 WHEN VALUE LIKE '%G' THEN TRIM(TRAILING 'G' FROM VALUE) * 1024 * 1024 * 1024 WHEN VALUE LIKE '%M' THEN TRIM(TRAILING 'M' FROM VALUE) * 1024 * 1024 WHEN VALUE LIKE '%K' THEN TRIM(TRAILING 'K' FROM VALUE) * 1024 WHEN REGEXP_LIKE(VALUE, '^[1-9]+[0-9]*$') = TRUE THEN TO_NUMBER(VALUE) END AS value FROM V$PARAMETER WHERE name = 'ARCH_CLEAN_UPPER_THRESHOLD'), usable_pct AS (SELECT TO_CHAR(100 * (b.total_blocks / a.value), '99.99') AS USABLE_PCT FROM arch_clean_upper_threshold a, value_stats b), space_limit AS (SELECT round(value/1024/1024/1024,2) AS SPACE_LIMIT FROM arch_clean_upper_threshold), number_of_files AS (SELECT COUNT(1) AS NUMBER_OF_FILES FROM V$ARCHIVED_LOG), space_used AS (SELECT ROUND(SUM(BLOCK_SIZE * BLOCKS)/1024/1024/1024,2) AS SPACE_USED FROM V$ARCHIVED_LOG), space_reclaimable AS (SELECT ROUND(SUM(b.size)/1024/1024/1024,2) AS SPACE_RECLAIMABLE FROM (SELECT REGEXP_SUBSTR(RCY_POINT, '[^-]+', 1, 2, 'i') AS value FROM v$database) a,(SELECT SEQUENCE# AS value, BLOCK_SIZE * BLOCKS AS size FROM V$ARCHIVED_LOG) b WHERE a.value > b.value), archive_dest AS (select value AS ARCHIVE_DEST from V$SYSTEM_PARAMETER where name='ARCHIVE_LOCAL_DEST') select USABLE_PCT,SPACE_LIMIT,NUMBER_OF_FILES,SPACE_USED,SPACE_RECLAIMABLE,ARCHIVE_DEST FROM usable_pct,space_limit,number_of_files,space_used,space_reclaimable,archive_dest;`	
Database Archive Logs	Query `v$archived_log` view: `select NAME, SEQUENCE# AS SEQUENCE, to_char(FIRST_TIME, 'yyyy-mm-dd hh24:mi:ss') as FIRST_TIME, to_char(NEXT_TIME, 'yyyy-mm-dd hh24:mi:ss') as NEXT_TIME, to_char(COMPLETION_TIME, 'yyyy-mm-dd hh24:mi:ss') as COMPLETION_TIME, BLOCKS, BLOCK_SIZE, COMPRESSED, FAL from v$archived_log;`	

Performance Analysis

Check Item	Implementation	Notes
VM Conversion Rate	Query `v$vm` , `v$sysstat` views: `SELECT t1.SWAPPED_OUT_BLOCKS / t2.value AS RATE FROM ( SELECT SWAPPED_OUT_BLOCKS FROM v$vm ) t1, ( SELECT value FROM V$SYSSTAT WHERE NAME = 'VM ALLOC' ) t2;`	
TOP SQL	1. Average Time TOP10 SQL `SELECT round(CPU_TIME / 1000, 2) AS CPU_TIME, EXECUTIONS , round(ELAPSED_TIME / 1000, 2) AS ALL_ELAPSED_TIME , round(ELAPSED_TIME / 1000 / EXECUTIONS, 2) AS AVG_TIME , to_char(LAST_ACTIVE_TIME,`	

Check Item	Implementation	Notes

	`'YYYY-MM-DD HH24:MI:SS') AS LAST_TIME, SQL_ID , SQL_TEXT FROM v$sqlarea WHERE EXECUTIONS > 0 ORDER BY round(ELAPSED_TIME / 1000 / EXECUTIONS, 2) DESC LIMIT 10;`
	2. Buffer Gets TOP10 SQL
	`SELECT BUFFER_GETS, EXECUTIONS , round(BUFFER_GETS / EXECUTIONS, 2) AS GETS_PER_EXEC , round(ELAPSED_TIME / 1000, 2) AS ALL_ELAPSED_TIME , to_char(LAST_ACTIVE_TIME, 'YYYY-MM-DD HH24:MI:SS') AS LAST_TIME, SQL_ID , SQL_TEXT FROM v$sqlarea WHERE EXECUTIONS > 0 ORDER BY BUFFER_GETS DESC LIMIT 10;`
	3. Disk Reads TOP10 SQL
	`SELECT DISK_READS, EXECUTIONS , round(DISK_READS / EXECUTIONS, 2) AS READS_PER_EXEC , round(ELAPSED_TIME / 1000, 2) AS ALL_ELAPSED_TIME , to_char(LAST_ACTIVE_TIME, 'YYYY-MM-DD HH24:MI:SS') AS LAST_TIME, SQL_ID , SQL_TEXT FROM v$sqlarea WHERE EXECUTIONS > 0 ORDER BY DISK_READS DESC LIMIT 10;`
	4. Parse Calls TOP10 SQL
	`SELECT PARSE_CALLS, EXECUTIONS , round(PARSE_CALLS / EXECUTIONS, 2) AS CALLS_PER_EXEC , round(ELAPSED_TIME / 1000, 2) AS ALL_ELAPSED_TIME , to_char(LAST_ACTIVE_TIME, 'YYYY-MM-DD HH24:MI:SS') AS LAST_TIME, SQL_ID , SQL_TEXT FROM v$sqlarea WHERE EXECUTIONS > 0 ORDER BY round(ELAPSED_TIME / 1000 / EXECUTIONS, 2) DESC LIMIT 10;`	

High Frequency SQL	Query `v$sql` view: `select SQL_ID, SQL_TEXT, PLSQL_EXEC_TIME, EXECUTIONS from v$sql where EXECUTIONS >= 10000;`	
Database Historical Load	Query `sys.wrm$_database_instance` , `sys.wrh$_sysstat` , `sys.wrm$_snapshot` views: `WITH dbinfo AS (SELECT DISTINCT dbid FROM sys.wrm$_database_instance LIMIT 1), t1 AS (SELECT snap_id, value FROM SYS.wrh$_sysstat, dbinfo WHERE SYS.wrh$_sysstat.dbid = dbinfo.dbid AND stat_id = 604), t2 AS (SELECT snap_id, begin_interval_time + (end_interval_time - begin_interval_time) / 2 AS snap_time FROM SYS.wrm$_snapshot, dbinfo WHERE SYS.wrm$_snapshot.dbid = dbinfo.dbid) SELECT t1.value AS db_times, to_char(t2.snap_time, 'YYYY-MM-DD HH24:MI:SS') AS snap_time FROM t1, t2 WHERE t1.snap_id = t2.snap_id AND t2.snap_time >= TIMESTAMP('%s') AND t2.snap_time <= TIMESTAMP('%s');` First query snapshot events and DBTime increments via above SQL, then calculate mid-value and DBTime difference for final result	

Memory Pool Hit Rate	1. Current memory hit rate
	Query `v$sysstat` view:
	`select (sum(decode(NAME, 'BUFFER GETS', VALUE, 0)) + sum(decode(NAME, 'BUFFER CR GETS', VALUE, 0)) - sum(decode(NAME, 'DISK READS', VALUE, 0))) / (sum(decode(NAME, 'BUFFER GETS', VALUE, 0)) + sum(decode(NAME, 'BUFFER CR GETS', VALUE, 0))) * 100 AS HIT_RATE FROM v$sysstat;`
	2. Historical memory hit rate
	Query `sys.wrm$_database_instance` , `sys.wrh$_sysstat` , `sys.wrm$_snapshot` views:
	`WITH dbinfo AS (SELECT DISTINCT dbid FROM SYS.wRM$_database_instance LIMIT 1), dbstat AS (SELECT snap_id, value, stat_id FROM SYS.wrh$_sysstat, dbinfo WHERE SYS.wrh$_sysstat.dbid = dbinfo.dbid), t1 AS (SELECT snap_id, value AS b_cr_get FROM dbstat WHERE stat_id = 120), t2 AS (SELECT snap_id, value AS b_buf_get FROM dbstat WHERE stat_id = 121), t3 AS (SELECT snap_id, value AS e_phy_read FROM dbstat WHERE stat_id = 131), t4 AS (SELECT t1.snap_id , (t1.b_cr_get + t2.b_buf_get) / (t1.b_cr_get + t2.b_buf_get + t3.e_phy_read) * 100 AS hit_rate FROM t1 JOIN t2 ON t1.snap_id = t2.snap_id JOIN t3 ON t1.snap_id = t3.snap_id), t5 AS (SELECT snap_id, begin_interval_time + (end_interval_time - begin_interval_time) / 2 AS snap_time FROM SYS.wrm$_snapshot, dbinfo WHERE SYS.wrm$_snapshot.dbid = dbinfo.dbid) SELECT to_char(t5.snap_time, 'YYYY-MM-DD HH24:MI:SS') AS snap_time, t4.hit_rate FROM t4 JOIN t5 ON t4.snap_id = t5.snap_id WHERE t5.snap_time >= TIMESTAMP('%s') AND t5.snap_time <= TIMESTAMP('%s') ORDER BY t5.snap_time;`	

Performance Config Check	1. Huge Pages Execute command: `grep -oP '(?<=\[).+?(?=\])' /sys/kernel/mm/transparent_hugepage/enabled` 2. Swap Memory Call `gopstuil` package's `mem.SwapMemory()`	

Lock Wait	1. Row lock wait
	`select count(*) as ROW_LOCK_WAIT_COUNT from v$lock where REQUEST in ('ROW');`
	2. Table lock wait
	`select count(*) as TABLE_LOCK_WAIT_COUNT from v$lock where REQUEST in ('TS','TX');`	

Long Transaction	Query `v$transaction` , `v$session` views: `select t.XID, to_char(t.START_DATE, 'yyyy-mm-dd hh24:mi:ss') as START_DATE, t.STATUS , t.RESIDUAL,`	

Check Item	Implementation	Notes
	`s.USERNAME, t.SID, t.USED_UBLK from v$transaction t, v$session s where t.START_DATE < sysdate - 3 / 24 and t.SID = s.SID;`	

object-check

Object Count Statistics

Check Item	Implementation	Notes

Object Total	1. Total object count
	`select count(*) as total_count from dba_objects;`
	2. Object count by owner
	`SELECT owner, object_type, COUNT(*) AS owner_object_count FROM dba_objects WHERE owner NOT IN ('SYS', 'SYSTEM') AND object_type NOT LIKE 'BIN$%' GROUP BY owner, object_type ORDER BY owner,object_type;`
	3. Object count by tablespace
	`SELECT tablespace_name, COUNT(*) AS tablespace_object_count FROM dba_segments WHERE segment_type IN ('TABLE', 'INDEX', 'VIEW', 'SEQUENCE') GROUP BY tablespace_name ORDER BY tablespace_name;`	

Object Status Check

Check Item	Implementation	Notes
Invalid Object List	Query `dba_objects` view: `select OBJECT_ID, OWNER, OBJECT_NAME, OBJECT_TYPE, STATUS from dba_objects where STATUS = 'INVALID';`	
Invisible Index List	Query `dba_indexes` view: `select OWNER,INDEX_NAME,VISIBILITY from dba_indexes where VISIBILITY !='VISIBLE';`	
Disabled Constraint List	Query `dba_constraints` view: `select OWNER,CONSTRAINT_NAME,CONSTRAINT_TYPE,STATUS from dba_constraints where STATUS ='DISABLED';`	

Tables

Check Item	Implementation	Notes
Tables with Too Many Columns	Query `dba_tab_cols` view: `select OWNER,TABLE_NAME,count(*) as COLUMN_COUNT from dba_tab_cols group by OWNER,TABLE_NAME having count(*)>80;`	
Tables with Too Many Indexes	Query `dba_indexes` view: `select TABLE_OWNER,TABLE_NAME,count(*) as INDEX_COUNT from dba_indexes group by TABLE_OWNER,TABLE_NAME having count(*) >8;`	
Partitioned Tables Without Partition Index	Query `dba_indexes` , `dba_part_key_columns` , `dba_part_tables` views: `SELECT b.OWNER ,b.name,a.PARTITIONING_TYPE ,b.tab_cols from (SELECT owner,TABLE_NAME,PARTITIONING_TYPE FROM DBA_PART_TABLES) a, (SELECT OWNER ,NAME ,LISTAGG(COLUMN_NAME,',') WITHIN group(ORDER BY COLUMN_POSITION) AS tab_cols FROM DBA_PART_KEY_COLUMNS WHERE OBJECT_TYPE ='TABLE' GROUP BY OWNER ,NAME) b WHERE a.OWNER = b.owner AND a TABLE_NAME =b.name AND a.owner<>'SYS' minus (...)`	
Tables with Row Size Exceeding Block Size	Query `dba_tab_columns` view: `SELECT a.OWNER, a.TABLE_NAME FROM (SELECT OWNER, TABLE_NAME, SUM(DATA_LENGTH) AS MAX_DL FROM DBA_TAB_COLUMNS WHERE OWNER <> 'SYS' AND DATA_TYPE NOT LIKE '%LOB' GROUP BY OWNER, TABLE_NAME) a, (SELECT to_number(decode(value, '8K', '8192', '16K', '16384', '32K', '32768', value)) AS VALUE FROM v$parameter WHERE NAME = 'DB_BLOCK_SIZE') b WHERE a.max_dl > b.value;`	

Check Item	Implementation	Notes
Hash Partitioned Tables with Non-Power-of-2 Partitions	Query `dba_part_tables` view: `select OWNER,TABLE_NAME,PARTITIONING_TYPE,PARTITION_COUNT from dba_part_tables where PARTITIONING_TYPE ='HASH' and abs(floor(log(2, PARTITION_COUNT)))!=log(2, PARTITION_COUNT) or log(2, PARTITION_COUNT)='Nan';`	
Tables with Case-Sensitive Names or Keywords/Special Characters	Query `all_tables` view: `SELECT owner, table_name FROM all_tables WHERE REGEXP_LIKE(table_name, '[^A-Z0-9_$#]') OR table_name IN (SELECT keyword FROM v$reserved_words WHERE reserved = 'Y') ORDER BY owner, table_name;`	
Columns with Case-Sensitive Names or Keywords/Special Characters	Query `all_tables` view: `SELECT t.owner, t.table_name, c.column_name FROM all_tab_columns c JOIN all_tables t ON c.owner = t.owner AND c.table_name = t.table_name WHERE REGEXP_LIKE(c.column_name, '[^A-Z0-9_$#]') OR c.column_name IN (SELECT keyword FROM v$reserved_words WHERE reserved = 'Y') ORDER BY t.owner, t.table_name, c.column_name;`	

Constraints

Check Item	Implementation	Notes
Child Table Foreign Keys Without Index	Query `dba_constraints` , `all_cons_columns` , `dba_ind_columns` , `dba_indexes` views: `WITH t1 AS (SELECT owner, CONSTRAINT_NAME, TABLE_NAME , LISTAGG(COLUMN_NAME, ',') WITHIN GROUP (ORDER BY posi) AS col_lst FROM (SELECT b.OWNER, b.CONSTRAINT_NAME, b.TABLE_NAME, b.COLUMN_NAME, b.posi FROM (SELECT OWNER, CONSTRAINT_NAME, TABLE_NAME FROM DBA_CONSTRAINTS WHERE CONSTRAINT_TYPE = 'R') a, (SELECT b.OWNER, b.CONSTRAINT_NAME, b.TABLE_NAME, b.COLUMN_NAME, b."POSITION" AS posi FROM ALL_CONS_COLUMNS b) b WHERE a.owner = b.OWNER AND a.CONSTRAINT_NAME = b.CONSTRAINT_NAME AND a.table_name = b.TABLE_NAME) GROUP BY owner, CONSTRAINT_NAME, TABLE_NAME), t2 AS (SELECT INDEX_OWNER, INDEX_NAME, TABLE_OWNER, TABLE_NAME , LISTAGG(COLUMN_NAME, ',') WITHIN GROUP (ORDER BY COLUMN_POSITION) AS ind_lst FROM DBA_IND_COLUMNS GROUP BY INDEX_OWNER, INDEX_NAME, TABLE_OWNER, TABLE_NAME) SELECT DISTINCT t1.owner, t1.CONSTRAINT_NAME, t1.TABLE_NAME, t1.col_lst FROM t1, t2 WHERE t1.owner <> 'SYS' AND t1.OWNER = t2.TABLE_OWNER AND t1.TABLE_NAME = t2.TABLE_NAME AND t1.col_lst <> t2.ind_lst;`	
Foreign Keys with Implicit Data Type Conversion	Query `dba_constraints` , `dba_cons_columns` , `dba_tab_columns` views: `WITH t1 AS (SELECT b.OWNER, b.CONSTRAINT_NAME, b.TABLE_NAME, b.COLUMN_NAME AS CHD_COL, b.posi , c.DATA_TYPE AS CHD_TYP FROM (SELECT OWNER, CONSTRAINT_NAME, TABLE_NAME FROM DBA_CONSTRAINTS WHERE CONSTRAINT_TYPE = 'R') a, (SELECT b.OWNER, b.CONSTRAINT_NAME, b.TABLE_NAME, b.COLUMN_NAME, b."POSITION" AS posi FROM DBA_CONS_COLUMNS b) b, (SELECT OWNER, TABLE_NAME, COLUMN_NAME, DATA_TYPE FROM DBA_TAB_COLUMNS) c WHERE a.owner = b.OWNER AND a.CONSTRAINT_NAME = b.CONSTRAINT_NAME AND a.table_name = b.TABLE_NAME AND b.OWNER = c.OWNER AND b.TABLE_NAME = c.TABLE_NAME AND b.COLUMN_NAME = c.COLUMN_NAME), t2 AS (...) SELECT t2.FK_OWNER, t2.FK_CON_NAME, t2.CHD_TAB, t1.CHD_COL, t1.CHD_TYP , t2.PRT_OWNER, t2.PRT_CON_NAME, t2.PRT_TAB, t2.PRT_COL, T2.PRT_TYP FROM t1, t2 WHERE t1.OWNER = t2.FK_OWNER AND t1.CONSTRAINT_NAME = t2.FK_CON_NAME AND t1.TABLE_NAME = t2.CHD_TAB AND t1.posi = t2.posi AND t1.CHD_TYP <> t2.PRT_TYP;`	

Indexes

Check Item	Implementation	Notes
Indexes with More Than 3 Levels	Query `dba_indexes` view: `select OWNER, INDEX_NAME, BLEVEL from dba_indexes where BLEVEL>3;`	
Indexes with Too Many Columns	Query `dba_ind_columns` view: `select INDEX_OWNER, INDEX_NAME, count(*) as column_count from dba_ind_columns group by INDEX_OWNER,INDEX_NAME having count(*) > 10;`	
Invisible Indexes	Query `dba_indexes` view: `select OWNER, INDEX_NAME, TABLE_OWNER, TABLE_NAME FROM dba_indexes where owner<> 'SYS' and`	

Check Item	Implementation	Notes
	`VISIBILITY <> 'VISIBLE';`	
Oversized Indexes	Query `dba_segments` view: `SELECT ind.OWNER AS ind_owner,ind.SEGMENT_NAME AS ind_name,ind.SEGMENT_TYPE as IND_SEGMENT_TYPE ,tab.SEGMENT_TYPE as TAB_SEGMENT_TYPE,tab.OWNER AS tab_owner ,tab.SEGMENT_NAME AS tab_name,ind.BYTES AS ind_bytes,tab.BYTES AS tab_bytes FROM DBA_SEGMENTS ind,DBA_SEGMENTS tab,DBA_INDEXES di WHERE IND.SEGMENT_TYPE IN ('INDEX','INDEX PARTITION') AND tab.SEGMENT_TYPE IN ('TABLE','TABLE PARTITION') AND ind.OWNER =di.OWNER AND ind.SEGMENT_NAME =di.INDEX_NAME AND tab.OWNER =di.TABLE_OWNER AND tab.SEGMENT_NAME =di.TABLE_NAME AND ind.BYTES > tab.BYTES;`	
Tables and Indexes in Different Schemas	Query `dba_indexes` view: `SELECT OWNER,INDEX_NAME ,TABLE_OWNER ,TABLE_NAME FROM dba_indexes WHERE OWNER <> TABLE_OWNER;`	

Sequences

Check Item	Implementation	Notes
Sequences Without Available Values	Query `dba_sequences` view: `SELECT SEQUENCE_OWNER ,SEQUENCE_NAME ,LAST_NUMBER / MAX_VALUE * 100 as USED_RATE FROM DBA_SEQUENCES ds WHERE LAST_NUMBER / MAX_VALUE > 7/10;`	

Jobs

Check Item	Implementation	Notes
Running Jobs	Query `dba_scheduler_jobs` view: `select OWNER ,JOB_NAME ,JOB_STYLE ,JOB_CREATOR ,JOB_ACTION from DBA_SCHEDULER_JOBS where STATE='RUNNING';`	

Packages

Check Item	Implementation	Notes
Package Body Without Package	Query `dba_source` view: `SELECT OWNER ,NAME, JOIN_STR FROM (SELECT OWNER ,NAME,LISTAGG(TYPE,'-') AS JOIN_STR FROM DBA_SOURCE GROUP by OWNER ,NAME) WHERE JOIN_STR<>'PACKAGE-PACKAGE BODY';`	

security-check

Login Check

Check Item	Implementation	Notes
Password Strength Control	Query `x$parameter` view: `SELECT VALUE FROM x$parameter WHERE name ='_CHECK_PASSWORD_COMPLEXITY';`	
Profiles Without Login Limit	Query `dba_profiles` view: `select PROFILE,RESOURCE_NAME ,RESOURCE_TYPE, LIMIT from DBA_PROFILES where PROFILE<>'DEFAULT' and RESOURCE_NAME='FAILED_LOGIN_ATTEMPTS' and LIMIT='UNLIMITED';`	

Audit Check

Check Item	Implementation	Notes
Audit Switch Check	Query `v$parameter` view: `SELECT VALUE FROM V$PARAMETER WHERE NAME='UNIFIED_AUDITING';`	

User and Privilege Check

Check Item	Implementation	Notes
Users Not in OPEN Status	Query `dba_users` view: `select username,ACCOUNT_STATUS from dba_users where ACCOUNT_STATUS!='OPEN';`	
Users with System Table Privileges	Query `dba_users` , `dba_tab_privs` views: `select GRANTEE , TABLE_NAME from DBA_TAB_PRIVS where OWNER='SYS' and TYPE='TABLE' and GRANTEE in (select username from dba_users);`	
All Users with DBA Role	Query `dba_role_privs` view: `select GRANTEE from dba_role_privs where GRANTED_ROLE='DBA';`	
Users with ALL PRIVILEGES|SYSTEM	Query `dba_users` , `dba_sys_privs` views: `select GRANTEE from dba_sys_privs where PRIVILEGE='ALL PRIVILEGES' AND GRANTEE IN ( SELECT USERNAME FROM DBA_USERS);`	
Users Using SYSTEM as Default Tablespace	Query `dba_users` view: `select username,default_tablespace from dba_users where default_tablespace = 'SYSTEM';`	

Audit Cleanup Task Details

Check Item	Implementation	Notes
Audit Cleanup Task Details	Query SQL: `select AUDIT_JOB.COUNT AS AUDIT_JOB_CLEAN_JOB, AUDIT_CLEAN_TIME.COUNT AS AUDIT_CLEAN_TIME_COUNT , AUDIT_RECORD.COUNT AS AUDIT_RECORD_COUNT from (SELECT COUNT(*) AS COUNT FROM DBA_AUDIT_MGMT_CLEANUP_JOBS) AS AUDIT_JOB, (select count(*) AS COUNT from DBA_AUDIT_MGMT_LAST_ARCH_TS) AS AUDIT_CLEAN_TIME , (select count(*) AS COUNT from UNIFIED_AUDIT_TRAIL) AS AUDIT_RECORD;`	

Audit File Size

Check Item	Implementation	Notes
Audit Cleanup Task Details	Query `dba_segments` view: `select segment_name ,bytes/1024/1024/1024 as size_gb from dba_segments where segment_name like '%AUD$%' and SEGMENT_TYPE = 'TABLE';`	

log-analysis

Error Log Analysis

Check Item	Implementation	Notes
Database Primary/Standby Connection Status	1. run.log error analysis Query `run.log` file, filter logs containing `errno` 2. alert.log error analysis Query `alert.log` file, filter alert information 3. Kernel error analysis Execute `dmesg` command, extract information containing keywords like "error", "warning", "warn", "failed", "invalid", "fault", "faulty", "timeout", "unable", "cannot", "corrupt", "corruption" 4. OS error log analysis Query `/var/log/messages` or `/var/log/syslog` file, filter logs containing keywords: "BIOS Error", "Error", "FAILED Result", "Hardware Error", "I/O error", "iptables denied", "refused connect", "Possible SYN flooding", "drop open reques", "connection reset", "error", "Out of memory", "Killed process"	

Database Change Log

Check Item	Implementation	Notes
Database Change Log	Query `run.log` file, filter `INFO` level logs	INFO level records key events during normal database operation, mainly including: Database status changes: e.g., startup, shutdown, primary/standby switch. Database key resource changes: e.g., tablespace, user additions/deletions. Database resource changes: e.g., thread start/stop. Database key activities: e.g., restart recovery, residual transactions. Database parameter changes: modifying system configuration, hidden parameters, etc.

Slow Log Analysis

Check Item	Implementation	Notes
Slow Log Parameters	Query `v$parameter` view	
Slow Log System Table	Query `sys.slow_log$` view	
Slow Log File	Query `slow.log` file	

REDO Log Analysis

Check Item	Implementation	Notes
REDO Log Analysis	Query `v$logfile` view: `select ID, NAME, STATUS, BLOCK_SIZE, BLOCK_COUNT, USED_BLOCKS, SEQUENCE# AS SEQUENCE from v$logfile;`	
REDO Log Count Analysis	Query `v$logfile` view: `select count(*) as total_count, SUM(CASE WHEN STATUS = 'CURRENT' THEN 1 ELSE 0 END) AS`	

Check Item	Implementation	Notes
	`current_count,SUM(CASE WHEN STATUS = 'ACTIVE' THEN 1 ELSE 0 END) AS active_count, SUM(CASE WHEN STATUS = 'INACTIVE' THEN 1 ELSE 0 END) AS inactive_count from v$logfile;`	

UNDO Log Analysis

Check Item	Implementation	Notes
Currently Used UNDO Space Size	Query `v$transaction`, `v$parameter` views: `SELECT round(a.USED_UBLK * b.value /1024/1024,3) AS SIZE_MB, XID from V$TRANSACTION as a , (SELECT VALUE FROM V$PARAMETER WHERE NAME='DB_BLOCK_SIZE') AS B ;`	
Excessive UNDO Block Usage	Query `v$transaction` view: `SELECT SUM(USED_UBLK) as TOTAL_BLOCK from V$TRANSACTION ;`	
Current Running Transactions and UNDO Usage	Query `v$transaction` view: `SELECT XID, SID,XRMID,XEXT, XNODE,XSN,STATUS,RESIDUAL, USED_UBLK, FIRST_UBAFIL,FIRST_UBABLK,FIRST_UBAVER ,FIRST_UBAREC,LAST_UBAFIL,LAST_UBABLK, PTX_XID, START_DATE,ISOLATION_LEVEL from V$TRANSACTION ;`	

report-interpretation

The health check report mainly includes health check overview information and 7 modules from different dimensions to measure system health, alon g with check results under each module.

Health Check Overview Information

Overview information mainly includes basic information about this check, score details, alert information, and check item information for each modu le with corresponding alert counts.

> The three color-coded numbers in the left menu indicate the number of alerts in check items under the current menu
> The left menu supports one-click collapse and expand, making it easy to quickly find check items with alerts

Overview Module

Contains two parts: Host Overview and Database Overview:

- Host Overview
  - Host Information
  - BIOS Information
- Database Overview

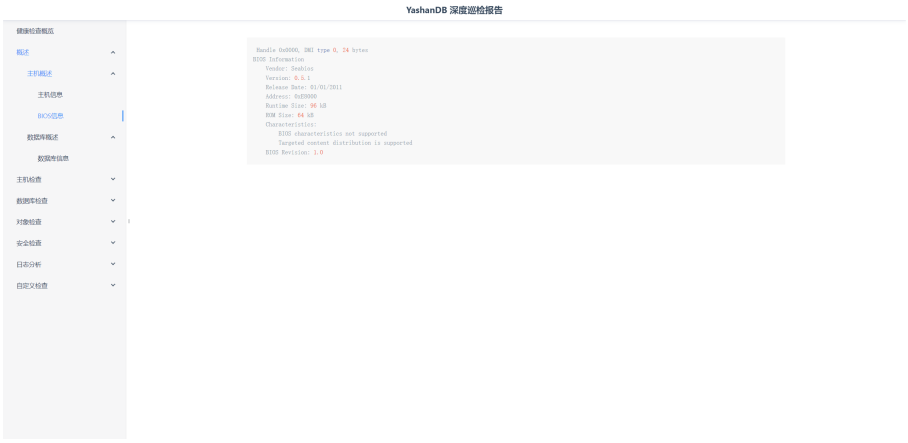
Host Overview

Host Information

Host information includes host operating system information, CPU information, network interface information, disk information, disk block device inf ormation, and memory information, displayed in chart form.

BIOS Information

BIOS information displays the current host's BIOS information, obtained by executing the dmidecode command on Linux system, with BIOS-related i nformation extracted and displayed.



Database Overview

Database Information

Database information records basic information about the database and instance, including database name, database version, database creation tim e, primary/standby role, deployment topology, instance status, instance startup time, etc.

Database file permissions show detailed permissions of database files on the current node, including file owner, group, and file permissions. If datab ase data file permissions are too high (other users have write permissions), an alert will be generated.

Host Check

Mainly includes host load module

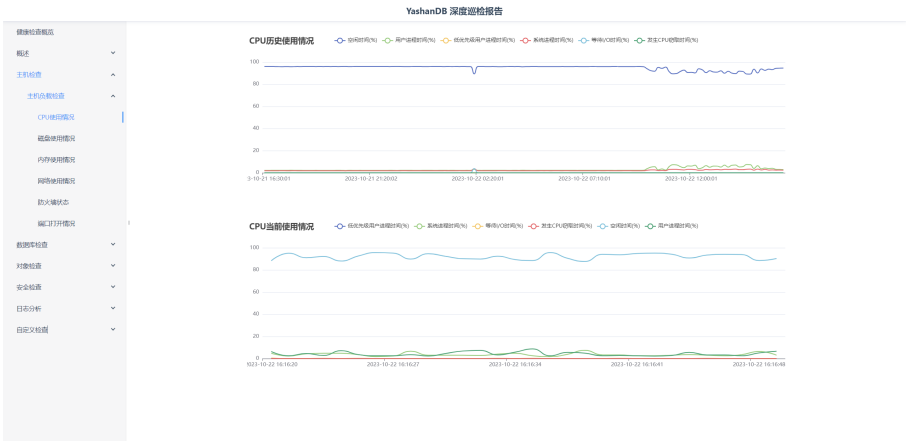
- Host Load Module
  - CPU Usage
  - Disk Usage
  - Memory Usage
  - Network Usage
  - Firewall Status
  - Port Open Status, etc.

Host Load Check

CPU Usage

CPU usage displays historical CPU load and current CPU load in curve chart form.

Historical CPU load shows CPU usage within the specified time range, such as idle time percentage, I/O wait time percentage, etc. Current CPU load s hows CPU usage during the check process.

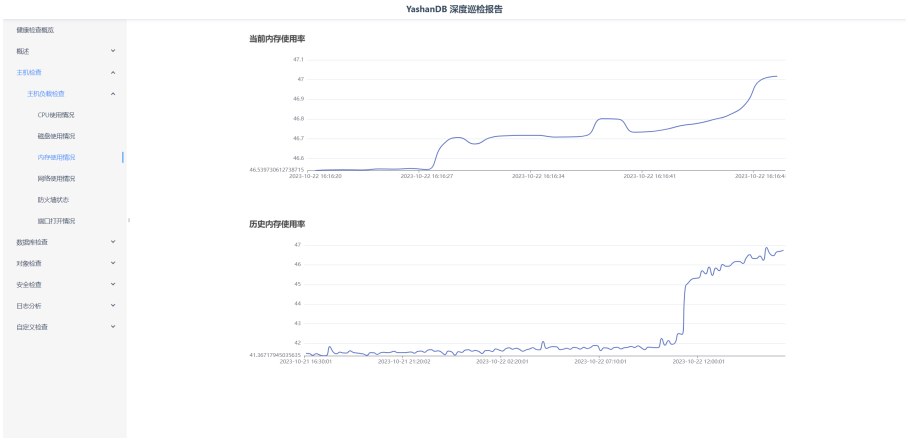


Disk Usage

Disk usage displays historical and current disk read/write data per second, tps, etc. in curve chart form.

Memory Usage

Memory usage displays current and historical memory usage in curve chart form.

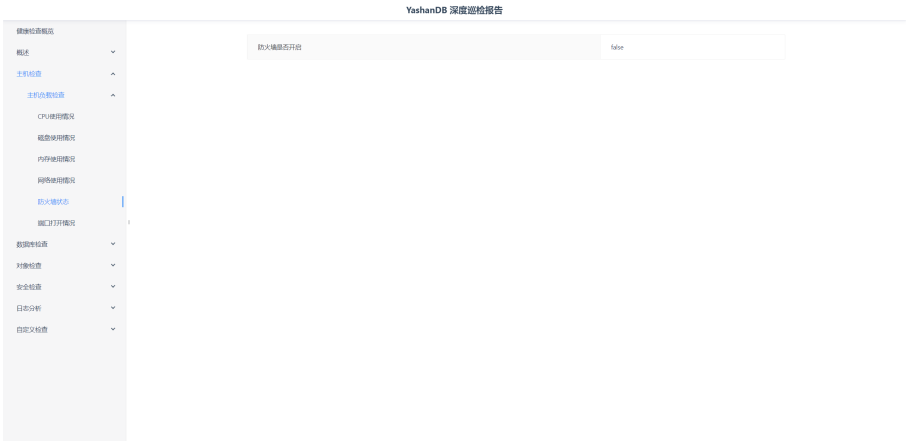


Network Usage

Network usage displays current and historical network received and sent data per second in curve chart form.

Firewall Status

Firewall status shows whether the firewall is enabled on the current machine. True means firewall is enabled, false means firewall is not enabled.



Port Open Status

Port open status shows the firewall rules of the current machine, obtained from the Linux command iptables -L.

Database Check

Mainly includes the following modules:

- Primary/Standby Check
  - Primary/Standby Connection Status
- Database Configuration Check
  - Database Parameter Check
  - OS Authentication Users
- Tablespace Check
  - Tablespace
  - Data Files
- Control File Check
  - Control File
- Backup Check
  - Backup Records
  - Full Backups Completed in Last 10 Days
  - Local Backup Set Path
- Load Check
  - Session Count Check
  - Shared Pool Check
- Archive Log Check
  - Database Archive Status
  - Database Archive Logs
- Performance Analysis
  - VM Conversion Rate
  - Wait Events
  - TOP10 SQL
  - High Frequency SQL
  - Database Historical Load
  - Memory Pool Hit Rate
  - Performance Configuration Check
  - Lock Wait
  - Long Transaction

Primary/Standby Check

Primary/Standby Connection Status

Primary/Standby check verifies the database primary/standby connection status, data from v$replication_status view.

> Only standby node views have this data, so if checking primary node, this data cannot be obtained

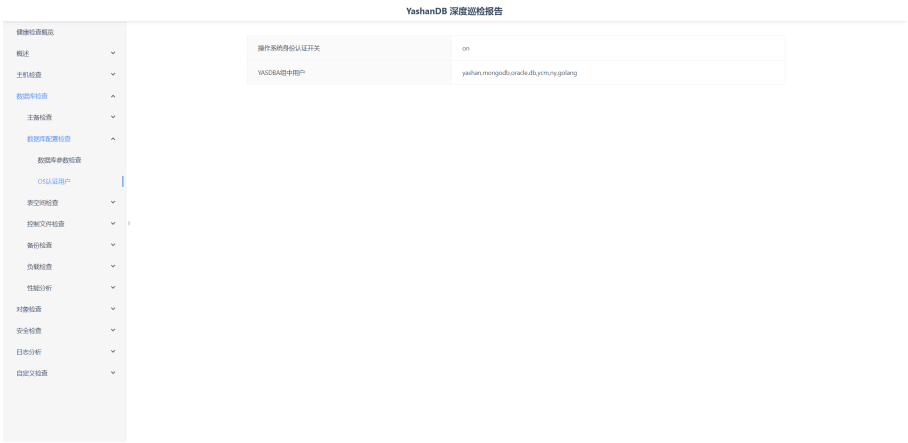
Database Configuration Check

Database Parameters

Database parameter check displays non-null parameters from v$parameter view in table form.

OS Authentication

OS authentication checks whether the current database has enabled operating system authentication. If enabled, it also displays all user information in the YASDBA group.



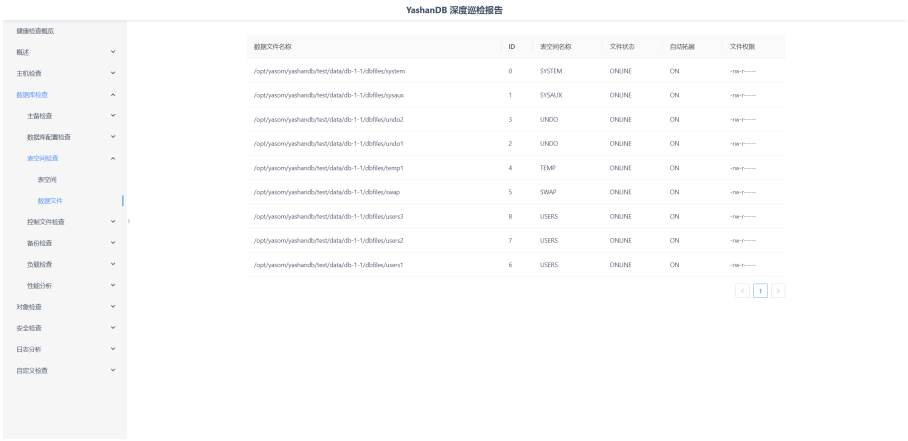
Tablespace Check

Tablespace

Tablespace page displays database tablespace related information in table form, including tablespace name, category, status, total bytes, usage rate, etc. It also alerts on tablespaces with auto-extension enabled.

Data Files

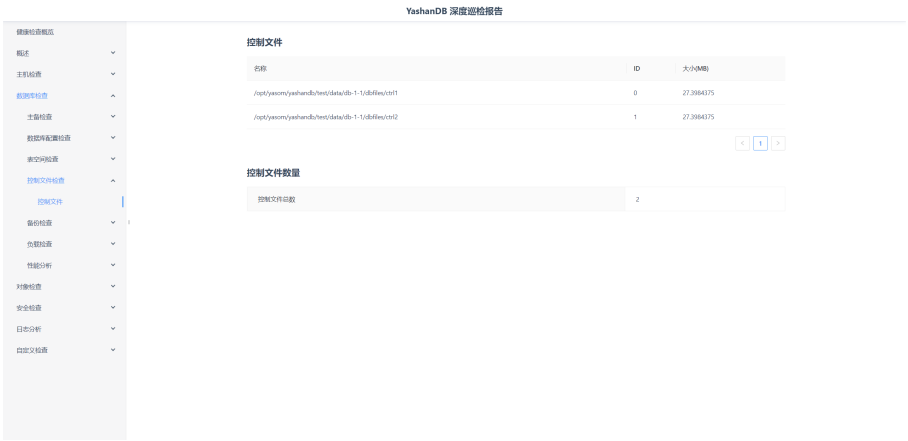
Data files page displays database data file related information in table form, including data file name, ID, tablespace name, status, auto-extension, etc.



Control File Check

Control File

Checks database control file information, such as control file name and size, and alerts on control file count. Too few control files may cause single point of failure leading to database unavailability.



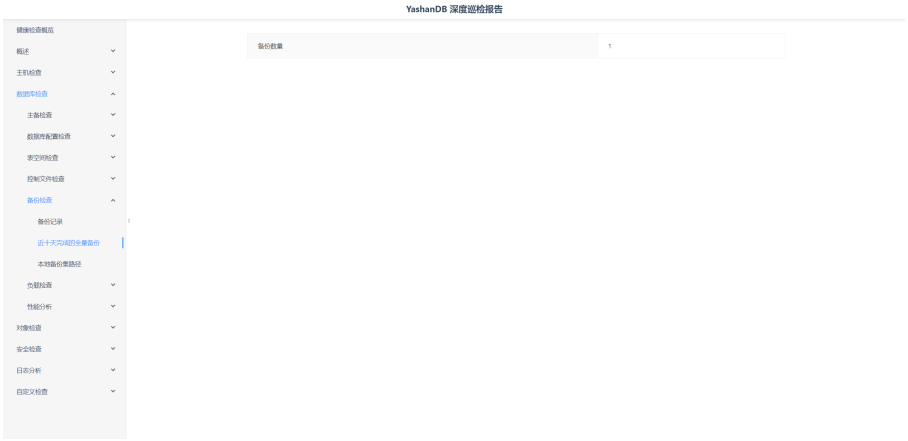
Backup Check

Database Backup Records

Backup records display backup set records in the database.

Full Backups Completed in Last 10 Days

Counts the number of full backup files completed in the last 10 days.



Local Backup Set Path

Local backup set path checks whether completed backups are saved in the local corresponding path. If not, an alert will be generated.

Load Check

Session Usage

Checks current system session count, user session count, max session count, total session count, session usage rate, etc.

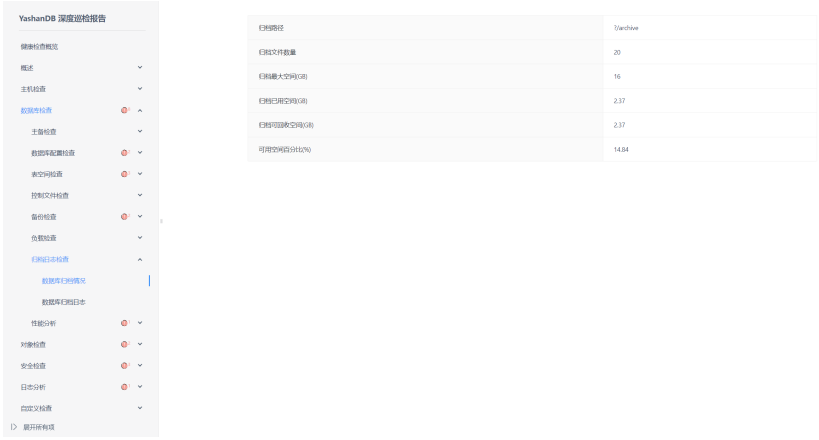
Shared Pool Information Check

Shared pool information checks database shared pool usage, including total shared pool size, free size, used size, and usage rate.

Archive Log Check

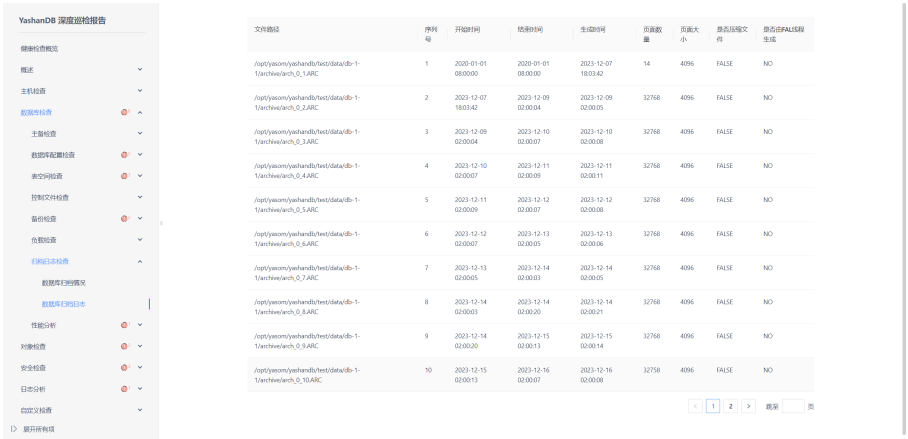
Database Archive Status

Database archive status displays database archive path, archive file space usage, etc.



Database Archive Logs

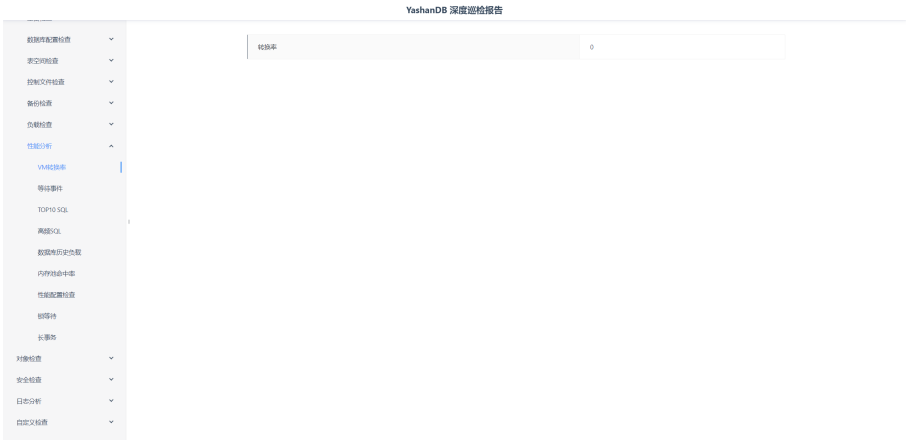
Database archive logs display related information for each archive log.



Performance Analysis

VM Conversion Rate

VM conversion rate displays current database VM conversion rate information.

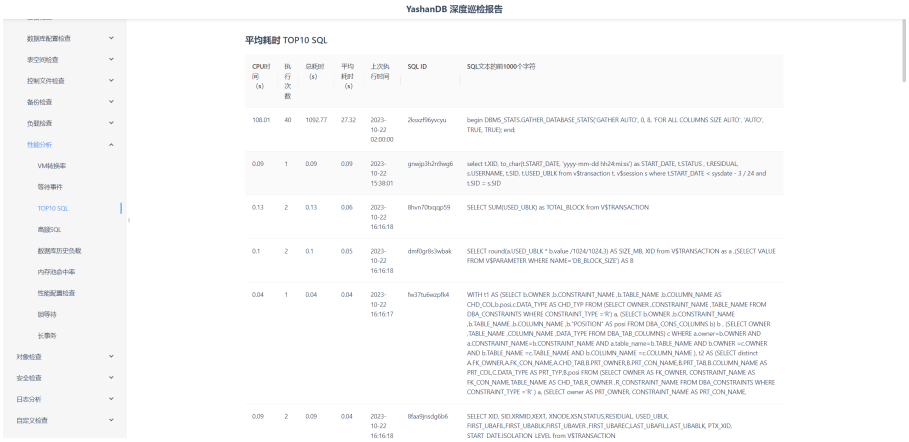


Wait Events

Wait events display TOP10 wait events within the check time range in table form, including event name, event category, etc.

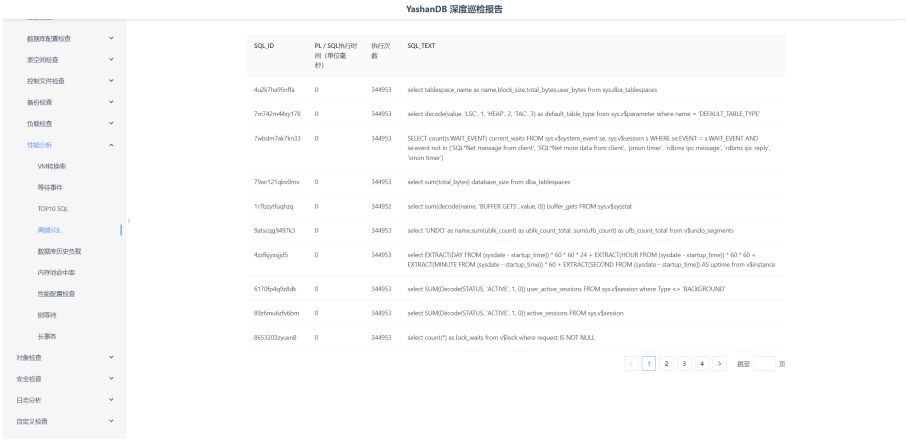
TOP10 SQL

TOP10 SQL displays detailed information of the 10 longest average execution time SQL statements, including SQL ID, first 1000 characters of SQL tex t, etc.



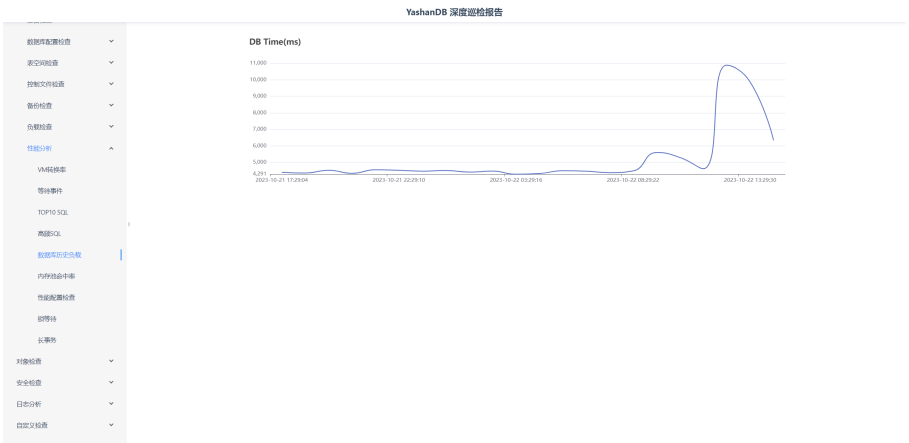
High Frequency SQL

High frequency SQL displays information of the 10 most frequently executed SQL statements.



Database Historical Load

Database historical load displays average DB Time within the specified time range in curve chart form.



Memory Pool Hit Rate

Memory pool hit rate displays database historical and current memory pool hit rate.

Performance Configuration Check

Performance configuration check checks whether database performance-related huge pages and swap memory configurations are enabled.

Lock Wait

Lock wait interface displays database row lock and table lock wait counts.

Long Transaction

Long transaction checks current database transactions with excessive execution time, including transaction ID, transaction status, etc.

Object Check

Object Count Statistics

Object Total

Statistics of database object total count, object count by each owner, and object count by each tablespace.

Object Status Check

Object check mainly checks tables, indexes, sequences, jobs, packages and other objects in the database, including the following modules:

Invalid Objects Check

Invalid object list displays detailed information of objects in invalid status, including object name, type, owner, etc.

Invisible Indexes

Invisible index list displays detailed information of indexes in invisible status, including index owner, index name, etc.

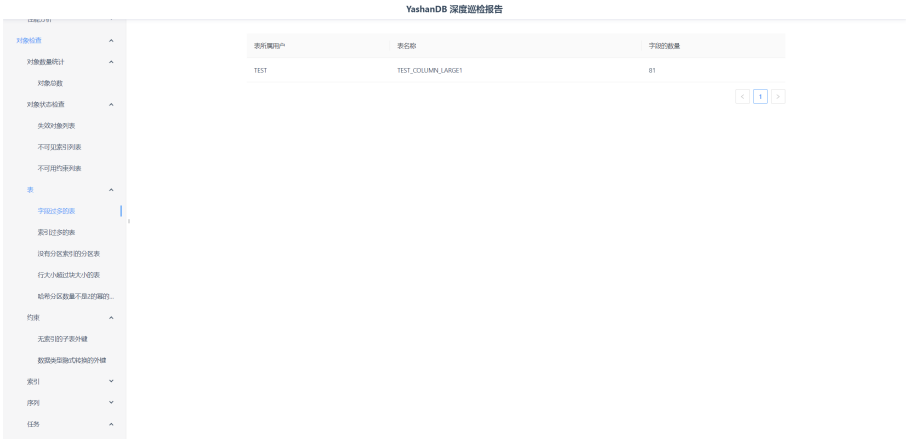
Disabled Constraints

Disabled constraint list displays detailed information of constraints in disabled status, including constraint owner, constraint name, constraint type, etc.

Tables

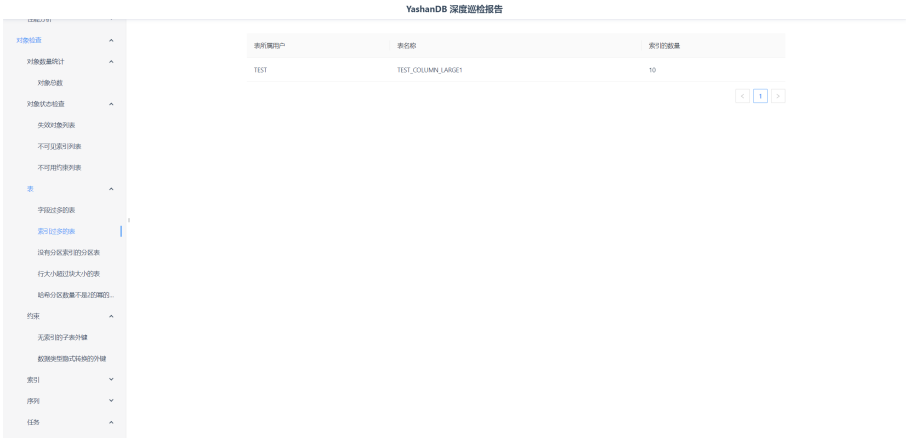
Tables with Too Many Columns

Tables with too many columns page displays detailed information of tables with more than 80 columns, including name, owner, and column count.



Tables with Too Many Indexes

Tables with too many indexes page displays detailed information of tables with more than 8 indexes, including name, owner, and index count.



Partitioned Tables Without Partition Index

Partitioned tables without partition index page displays detailed information of such tables, including name, owner, partition type, and partition key column information.

Tables with Row Size Exceeding Block Size

Tables with row size exceeding block size page displays name and owner information of such tables.

Hash Partitions with Non-Power-of-2 Count

Hash partitioned tables with non-power-of-2 partition count page displays detailed information of such tables, including name, owner, partition type, and partition count.

Constraints

Child Table Foreign Keys Without Index

Child table foreign keys without index interface displays detailed information of such tables, including parent table, child table, constraint name, and related column information.

Foreign Keys with Implicit Data Type Conversion

Since YashanDB does not support implicit data type conversion, this page is usually empty.

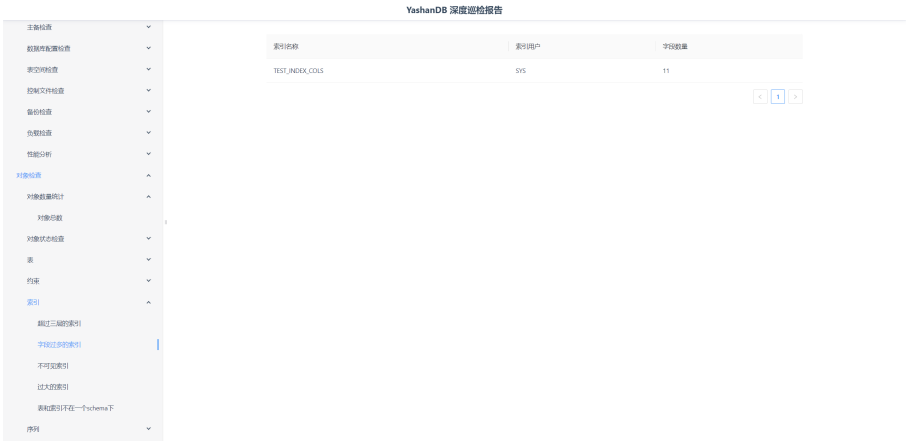
Indexes

Indexes with More Than 3 Levels

Mainly displays index information with more than three levels.

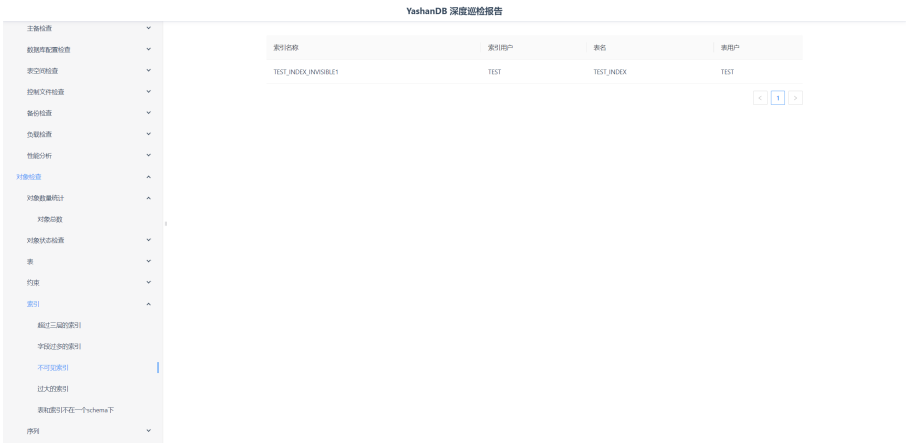
Indexes with Too Many Columns

Mainly displays indexes with too many columns.



Invisible Indexes

Mainly displays invisible index information.



Oversized Indexes

Mainly displays oversized index information.

Tables and Indexes in Different Schemas

Mainly displays index and table information where table and index are not in the same schema.

Sequences

Sequences Without Available Values

Mainly displays sequence information with usage rate exceeding 70%.

Jobs

Running Jobs

Mainly displays currently running database JOB information.

Packages

Package Body Without Package

Mainly displays package body without package.

Security Check

Security check mainly checks user login, user privileges, and auditing, including the following modules and check items:

- Login Check
  - Password Strength
  - Profiles Without Login Limit
- User and Privilege Check
  - Users Not in OPEN Status
  - Users with System Table Privileges
  - Users with DBA Role
  - Users with ALL PRIVILEGES|SYSTEM
  - Users with SYSTEM as Default Tablespace
- Audit Cleanup Task Details
- Audit File Size

Login Check

Password Strength

Profiles Without Login Limit

User and Privilege Check

Users Not in OPEN Status

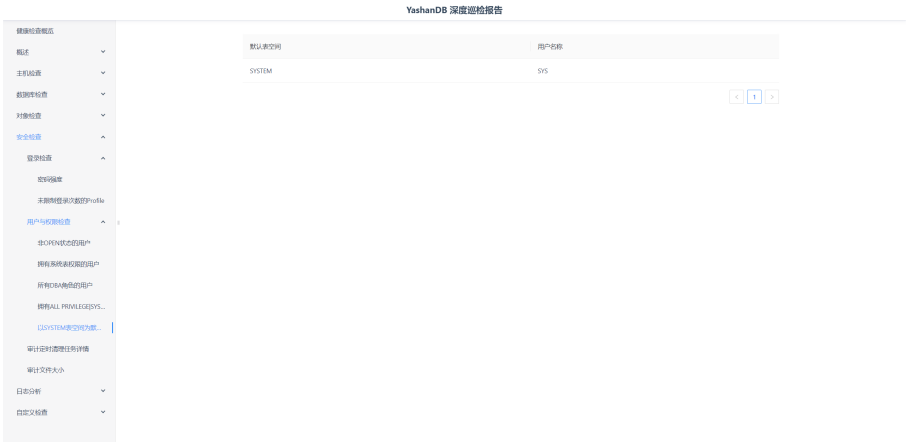
Users with System Table Privileges

Users with DBA Role

Users with ALL PRIVILEGES|SYSTEM

!1697966315349

Users with SYSTEM as Default Tablespace

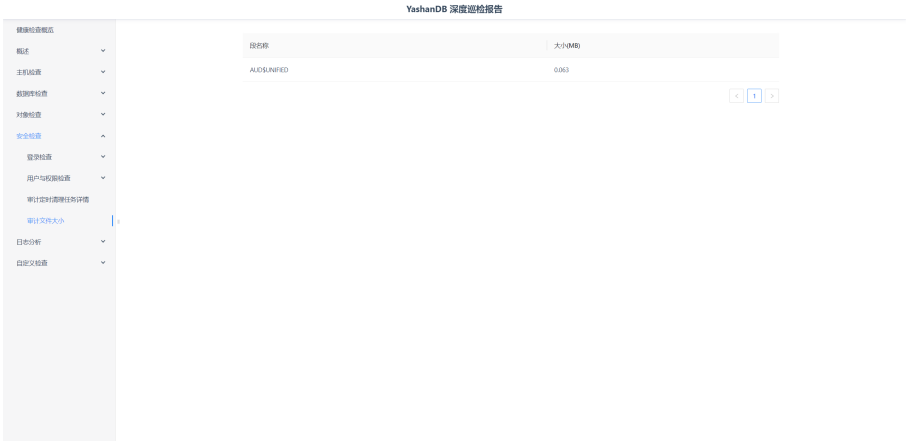


Audit Cleanup Task Details

Mainly displays audit cleanup task count, audit log cleanup time point count, and audit log count.

Audit File Size

Mainly displays segment information related to auditing.



Log Analysis

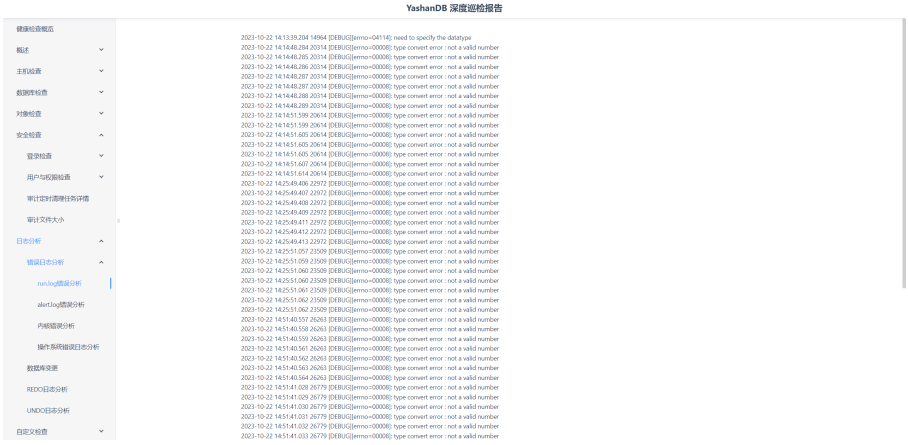
Log analysis module mainly filters and queries database logs, operating system logs, REDO, and UNDO logs, including:

- Error Log Analysis (queries logs matching specific conditions within time range)
  - run.log Error Analysis
  - alert.log Error Analysis
  - Kernel Error Analysis
  - OS Error Log Analysis
- Database Change Log (database change logs in run.log)
- REDO Log Analysis
- UNDO Log Analysis

Error Log Analysis

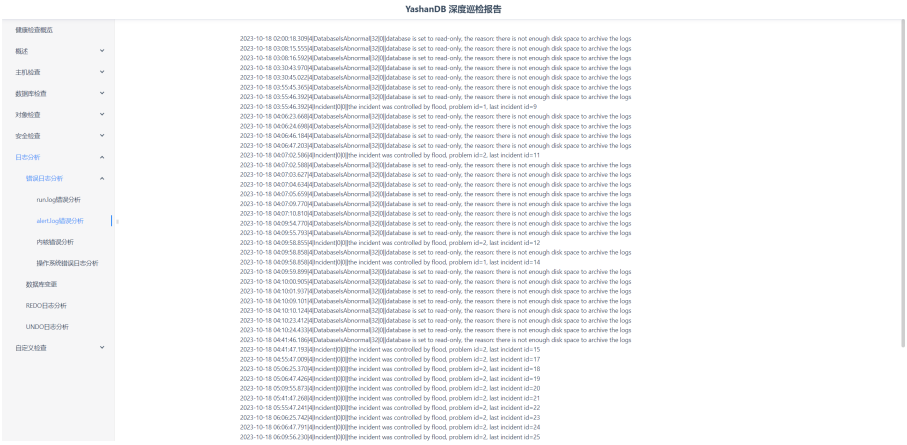
run.log Error Analysis

Mainly collects logs within time range with *errno* in log body.



alert.log Error Analysis

Mainly collects alert logs within time range with alert action of 0, i.e., alert log status is *reported*.



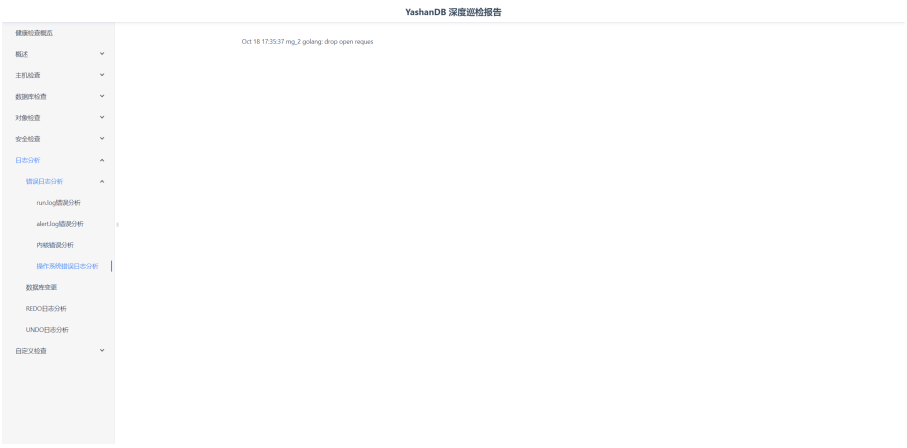
Kernel Error Analysis

Mainly collects dmesg logs containing the following keywords:

- error
- warning
- warn
- failed
- invalid
- fault
- faulty
- timeout
- unable
- cannot
- corrupt
- corruption

OS Error Log Analysis

Error logs from /var/log/messages or /var/log/syslog in the operating system.



Database Change Log

Database change logs from run.log.

Slow Log Analysis

Includes slow log parameters, slow log system table, and slow log file.

REDO Log Analysis

Includes REDO log file information and REDO log count analysis.

UNDO Log Analysis

Mainly includes currently used UNDO space size, excessive UNDO block usage, and current running transactions with UNDO usage.

configuration-file

Tool Configuration File: `{yhc_home}/config/yhc.toml`

```toml
log_level = "DEBUG"
max_duration = "30d"
min_duration = "1m"
range = "24h"
output = "./results"
language = "zh"
sql_timeout = 10
scrape_interval = 1
scrape_times = 30
metric_paths = ["./config/default_metric.toml", "./config/custom_metric.toml"]
default_module_path = "./config/report_module.toml"
evaluate_model_path = "./config/evaluate_model.toml"
network_io_discard = "^lo$,^veth.*,^virbr.*,^br.*,^tap.*,^tun.*,^docker.*,^flannel.*"
```

This is the YHC tool's main configuration file entry. The meaning of each field is as follows:

- log_level: Tool check information time range maximum value that affects -r or -s/-e functionality, default 30 days
- max_duration: Tool check information time range maximum value that affects -r or -s/-e functionality, default 30 days
- min_duration: Tool check information time range minimum value that affects -r or -s/-e functionality, default 1 minute
- range: Tool check time range, default is 24h before current time
- output: Main path where checked data is stored, default is in the results directory under tool execution directory
- language: Language used by the tool, supports "zh" (Chinese) and "en" (English), default is "zh". This setting affects CLI output, HTML report interface, and Word report content. Command line parameter `-l` or `--language` takes precedence over this configuration
- sql_timeout: AWR generation maximum timeout, default 10 seconds
- scrape_interval: Time interval for network load related data check, default 1s
- scrape_times: Total number of repetitions based on time interval for network load related data check, default 30 times
- metric_paths: Check metric configuration files, used to configure check metrics, default includes default metric configuration file and custom metric configuration file
- default_module_path: Report module configuration file, used to configure report display content
- evaluate_model_path: Score evaluation model configuration file, used to configure health check score evaluation model
- network_io_discard: Data discarded by default when querying server network load

Report Module Configuration File: `{yhc_home}/config/report_module.toml`

```toml
[[modules]]
  name = "overview"
  name_alias = "Overview"

  [[modules.children]]
    name = "overview_host"
    name_alias = "Host Overview"
    metric_names = ["host_info", "host_cpu_info", "host_disk_info", "host_disk_block_info", "host_bios_info",
"host_memory_info", "host_network_info"]

  [[modules.children]]
    name = "overview_yasdb"
    name_alias = "Database Overview"
    metric_names = ["yasdb_instance", "yasdb_database", "yasdb_file_permission", "yasdb_listen_address"]

[[modules]]
  name = "host_check"
  name_alias = "Host Check"

  [[modules.children]]
    name = "host_workload_check"
    name_alias = "Host Load Check"
    metric_names = ["host_history_cpu_usage", "host_current_cpu_usage"]
```

```
# ......
# More file content omitted, see configuration file for details
```

This is the YHC tool's report module configuration file, responsible for organizing report display content. The meaning of each field is as follows:

- modules: Defines report content for each module
  - name: Defines module name (English recommended)
  - name_alias: Defines module alias (for display in report, usually in user's language). If no alias is defined, module name is used
  - modules.children: Defines sub-modules of the module
    - name: Defines sub-module name (English recommended)
    - name_alias: Defines sub-module alias (for display in report). If no alias is defined, sub-module name is used
    - metric_names: Defines metric names under the module, i.e., which check metric information is displayed under the module

Default Metric Configuration File: `{yhc_home}/config/default_metric.toml`

```
[[metrics]]
  name = "yasdb_instance"
  name_alias = "Instance Information"
  module_name = "overview"
  metric_type = "sql"
  default = true
  enabled = true
  sql = "select status as instance_status, version, startup_time from v$instance;"
  [metrics.column_alias]
    INSTANCE_STATUS = "Database Instance Status"
    STARTUP_TIME = "Database Instance Startup Time"
    VERSION = "Database Version"
  [metrics.item_names]
    INSTANCE_STATUS = "instance_status"
  [metrics.alert_rules]

    [[metrics.alert_rules.critical]]
      expression = "instance_status != 'OPEN'"
      description = "Instance status abnormal"
      suggestion = "Recommend checking instance status"

[[metrics]]
  name = "yasdb_tablespace"
  name_alias = "Tablespace"
  module_name = "yasdb_check"
  metric_type = "sql"
  default = true
  enabled = true
  column_order = ["TABLESPACE_NAME", "CONTENTS", "STATUS", "ALLOCATION_TYPE", "USED_BYTES", "TOTAL_BYTES", "USED_RATE",
"DATA_PERCENTAGE"]
  number_columns = ["USED_RATE","USED_BYTES","TOTAL_BYTES"]
  labels = ["TABLESPACE_NAME"]
  sql = "SELECT TABLESPACE_NAME, CONTENTS, STATUS, ALLOCATION_TYPE, TOTAL_BYTES - USER_BYTES AS USED_BYTES, TOTAL_BYTES,
(TOTAL_BYTES - USER_BYTES) / TOTAL_BYTES * 100 AS USED_RATE FROM SYS.DBA_TABLESPACES;"
  [metrics.column_alias]
    TABLESPACE_NAME = "Tablespace Name"
    CONTENTS = "Tablespace Type"
    STATUS = "Tablespace Status"
    ALLOCATION_TYPE = "EXTEND Allocation Type"
    USED_BYTES = "Used Bytes"
    TOTAL_BYTES = "Total Bytes"
    USED_RATE = "Usage Rate(%)"
    DATA_PERCENTAGE = "Data Segment Usage Percentage"
  [metrics.item_names]
    STATUS = "tablespace_status"
    ALLOCATION_TYPE = "allocation_type"
    USED_RATE = "tablespace_used_rate"
  [metrics.alert_rules]

    [[metrics.alert_rules.warning]]
      expression = "tablespace_status != 'ONLINE'"
      description = "Tablespace status check"
```

```
      suggestion = "Tablespace status is abnormal, recommend checking tablespace status to confirm the cause"

    [[metrics.alert_rules.warning]]
      expression = "allocation_type == 'AUTO'"
      description = "Tablespace auto-extension enabled"
      suggestion = "Recommend disabling this feature to avoid transactions occupying too much UNDO causing disk space
unavailability"

    [[metrics.alert_rules.warning]]
      expression = "tablespace_used_rate >= 80 && tablespace_used_rate < 90"
      description = "Tablespace usage rate"
      suggestion = "Tablespace usage rate is high, recommend checking tablespace usage and reducing usage rate by clearing
space or adding data files. When tablespace is full, it may cause business suspension or database suspension"

    [[metrics.alert_rules.critical]]
      expression = "tablespace_used_rate >= 90"
      description = "Tablespace usage rate"
      suggestion = "Tablespace usage rate is high, recommend checking tablespace usage and reducing usage rate by clearing
space or adding data files. When tablespace is full, it may cause business suspension or database suspension"


# ......
# More file content omitted, see configuration file for details
```

This is the YHC tool's default metric configuration file, responsible for defining default check metrics. The meaning of each field is as follows:

- metrics: Defines content for each check metric
  - name: Defines check metric name (English recommended)
  - name_alias: Defines check metric alias (for display in report). If no alias is defined, check metric name is used
  - module_name: Module name the metric belongs to
  - default: Whether it is a default metric
  - enabled: Whether the metric is enabled by default. If `false`, the check metric will not be displayed when showing optional metrics
  - metric_type: Metric type, supports `sql` and `bash`. Default metrics don't need to specify metric type, mainly used for custom metrics
  - column_order: If check result is in table form, the order of each column can be specified through this field
  - metrics.column_alias: Used to configure column name aliases for check results. Report displays aliases first; if no alias, displays original column name
  - metrics.item_names: Used to configure alert metric names corresponding to column names. Alert metrics are basic elements of alert expressions
  - number_columns: Used to convert corresponding column's alert metric results to numbers. Number type alert metrics can perform mathematical operations; default alert metric results are strings
  - labels: Used to add labels to alert metrics. When check results have multiple columns, to distinguish each column, certain fields need to be specified as unique identifiers. E.g., in `dba_tablespaces` view, unique identifier field is: TABLESPACE_NAME
  - sql: Used to configure SQL query statement for checking corresponding metric. `sql` type metrics require this
  - metrics.alert_rules: Used to define check item alerts and suggestions if needed
    - metrics.alert_rules.info: Used to define info level alerts
      - expression: Used to define alert expression
      - description: Used to define alert description
      - suggestion: Used to define alert suggestion
    - metrics.alert_rules.warning: Used to define warning level alerts
      - Sub-fields can refer to `metrics.alert_rules.info`
    - metrics.alert_rules.critical: Used to define critical level alerts
      - Sub-fields can refer to `metrics.alert_rules.info`

Health Check Score Evaluation Model Configuration File: `{yhc_home}/config/evaluate_model.toml`

```
# Used to evaluate check results
total_score = 100  # Total score
default_metric_weight = 5 # Default weight for metrics not explicitly specified in metrics_weight field
max_alert_total_weight = 10 # Total weight of alerts generated by single metric
ignore_same_alert = false # Whether single metric same level alerts only deduct points once. E.g., if a metric generates ten
critical alerts, only deduct critical alert score once
ignore_failed_metric = true # Whether to ignore failed check metric items

[metrics_weight] # Explicitly specify weight for specific metrics
```

```
  # Category 1 metrics
  yasdb_database = 30
  host_disk_info = 30
  yasdb_replication_status = 30
  yasdb_controlfile_count = 30

  # Category 2 metrics
  yasdb_tablespace = 20
  yasdb_datafile = 20
  yasdb_session = 20
  yasdb_undo_size = 20

  # Category 3 metrics

  # Category 4 metrics
  host_huge_page = 7
  host_swap_memory = 7
  yasdb_security_user_use_system_tablespace = 7
  yasdb_redo_log_count = 7

  # Category 5 metrics
  yasdb_file_permission = 5
  yasdb_parameter = 5
  yasdb_os_auth = 5
  yasdb_controlfile = 5
  yasdb_security_password_strength = 5
  yasdb_security_maximum_login_attempts = 5
  yasdb_security_audit_cleanup_task = 5

[alerts_weight] # Specify alert deduction weight. Formula for metric alert deduction: Single metric score * alert deduction
weight / single metric alert total weight
  critical = 3
  warning = 2
  info = 1


[health_model] # Health model, health status corresponding to scores
  [health_model.excellent]
    min = 95
    max = 100
  [health_model.good]
    min = 80
    max = 95
  [health_model.fair]
    min = 70
    max = 80
  [health_model.poor]
    min = 60
    max = 70
  [health_model.critical]
    min = 0
    max = 60

[health_status_alias]
  critical = "Critical"
  excellent = "Excellent"
  fair = "Fair"
  good = "Good"
  poor = "Poor"
```

This is the YHC tool's health check score evaluation model configuration file, responsible for defining score evaluation model. The meaning of each fi eld is as follows:

- total_score: Total score for health check
- default_metric_weight: Default weight when metric weight is not explicitly specified
- max_alert_total_weight: Maximum total weight of alerts generated by single metric
- ignore_same_alert: Whether same level alerts of single metric only deduct points once. E.g., when this field is true, if a metric generates 10 critical l evel alerts, only deduct critical level alert score once
- ignore_failed_metric: Whether to ignore failed metrics. E.g., when this field is true, failed check metrics will not participate in score calculation

- metrics_weight: Used to explicitly specify weight for specific metrics
- module_weight: Used to batch specify weight for all metrics under a module
- alerts_weight: Specify weight for specific alert levels, used to calculate score deduction for corresponding alert levels
- health_model: Used to specify health model, i.e., database health status corresponding to different scores
  - health_model.excellent: Score range corresponding to excellent status, i.e., closed interval [min, max]
    - min: Minimum score
    - max: Maximum score
  - ......
- health_status_alias: Alias mapping for health status

YHC tool's health check score evaluation model uses deduction system. Final score is sum of actual scores of each metric.

```
Each metric actual score = Metric total score - Metric alert deductions

Metric total score = Metric weight / Total weight of all participating metrics * Total score

Metric alert deductions = Sum of each alert's weight / Metric total weight * Metric total score

(Note: Metric alert deductions maximum is single metric total score, i.e., each metric actual score minimum is 0, no negative
scores)
```

Multi-Node Check Configuration File: `{yhc_home}/config/nodes_config.toml`

```
# you can add nodes you want check just like this:
# that means you want check the node whose listen_addr is '127.0.0.1:1688'
# [[nodes]]
#   listen_addr = "127.0.0.1:1688"


# you can also check nodes in other clusters if you fill the user and password
#
# [[nodes]]
#   listen_addr = "127.0.0.1:1688"
#   user = "test"
#   password = "test"
```

This is the YHC tool's multi-node check configuration file. Users can configure this file to specify nodes to check in multi-node check mode. Field meanings are as follows:

- listen_addr: Required, listen address of node to check
- user: Optional, can specify database user for checking current node. If not specified, uses username/password from command line parameters or interactive interface
- password: Optional, can specify database user password for checking current node. If not specified, uses username/password from command line parameters or interactive interface

This is the YHC tool's custom metric configuration file. If you need custom check metrics, you can add them here. Please refer to Custom Check Items for guidance.

All above configuration items can be adjusted according to specifications and take effect on next execution

alert-expression

Introduction to Alert Expression

Alert expressions are used to define alert conditions. Metrics and operators are the basic elements of alert expressions, and metrics + operators for m alert expressions.

Metric

A metric can consist of three parts: metric name, metric value, and metric labels, where metric labels are optional.

- Metric name: Starts with a letter, followed by letters, underscores, or numbers. Example: cpu_usage (CPU usage rate).
- Metric value: The data type of metric value is basic type, currently supporting numbers and strings, where strings are wrapped in single quotes. Ex ample: cpu_usage == 0.1 (CPU usage equals 0.1), hostname == 'localhost' (hostname is localhost).
- Metric labels: When a metric name has multiple data items, labels can be used to identify additional attributes of metric items. Labels are string ty pe. Example: tablespace_usage{name == 'UNDO'} (usage rate of tablespace named UNDO).

Operator

Currently supports the following three types of operators:

- Logical operators: [&&, ||]
- Comparison operators: [>, >=, <, <=, ==, !=, ~=, in]
- Arithmetic operators: [+, -, *, /, %]

Common operators work similar to their original meanings. Here's an explanation of two special operators:

- ~=: Used for regex matching strings, e.g., firewall_status ~= '.*active', tablespace_usage{name ~= '^U.+'} > 0.5
- in: Used to check if a string is in a list, e.g., firewall_status in ['active', 'inactive'], tablespace_usage{name in ['UNDO', 'USER']} > 0.5

Defining Alert Expressions

Direct explanation may be abstract, so let's learn through examples.

Single Metric Alert

Single Metric Alert Without Labels

1. CPU usage alert, assuming metric name is: cpu_usage. Metric value type is number, here are some alert examples:

Alert Expression	Alert Condition
cpu_usage > 0.75	CPU usage greater than 75%
cpu_usage > 0.75 && cpu_usage <= 0.9	CPU usage greater than 75% and less than or equal to 90%
cpu_usage > 0.75 || cpu_usage <= 0.001	CPU usage greater than 75% or less than or equal to 0.1%

2. Firewall status alert, assuming metric name is: firewall_status. Metric value type is string, here are some alert examples:

Alert Expression	Alert Condition
firewall_status != 'active'	Firewall status is not active
firewall_status == 'inactive'	Firewall status is inactive
firewall_status == 'inactive' || firewall_status == 'dead'	Firewall status is inactive or dead
firewall_status in ['inactive', 'dead']	Firewall status is inactive or dead
firewall_status ~= '*.active'	Firewall status matches any characters followed by active

Additional notes:

- in operator is used to check if a string value is in a string list, where the string list is wrapped in square brackets and strings are separated by com mas.
- ~= operator is used for regex matching strings, matching means condition is met.

Single Metric Alert With Labels

1. Tablespace usage alert, assuming metric name is: tablespace_usage. Metric value type is number, with one label: tablespace name (name), here are some alert examples:

Alert Expression	Alert Condition
tablespace_usage > 0.8	All tablespace usage greater than 80%
tablespace_usage{name == 'UNDO'} > 0.9 && tablespace_usage{name == 'UNDO'} < 0.95	UNDO tablespace usage greater than 90% and less than or equal to 95%
tablespace_usage{name != 'UNDO'} > 0.7	Non-UNDO tablespace usage greater than 70%
tablespace_usage{name == 'UNDO'} > 0.7 || tablespace_usage{name !='UNDO'} > 0.8	UNDO tablespace usage greater than 70% or other tablespace usage greater than 80%
tablespace_usage{name != 'UNDO' && name!='DATA'} > 0.7	Non-UNDO and non-DATA tablespace usage greater than 70%
tablespace_usage{name ~= '^U.*'} > 0.7	Tablespace names starting with U usage greater than 70%
tablespace_usage{name == 'UNDO' || name == 'DATA'} > 0.7	Tablespace named UNDO or DATA usage greater than 70%
tablespace_usage{name in ['UNDO', 'DATA']} > 0.7	Tablespace named UNDO or DATA usage greater than 70%

Multiple label alerts are similar, for example:

Alert Expression	Alert Condition
metric_name{label1 == 'value1' || label2 == 'value2'} > 1	Metric value greater than 1 where label1 equals value1 or label2 equals value2
metric_name{label1 == 'value1' && label2 == 'value2'} > 1	Metric value greater than 1 where label1 equals value1 and label2 equals value2

Multiple Metric Combined Alert

Multiple Metric Alert Without Labels

1. CPU usage and memory usage combined alert. Assuming metric names are: cpu_usage, memory_usage, here are some alert examples:

Alert Expression	Alert Condition
cpu_usage > 0.75 || memory_usage > 0.7	CPU usage greater than 75% or memory usage greater than 70%
cpu_usage > 0.75 && memory_usage > 0.7	CPU usage greater than 75% and memory usage greater than 70%
cpu_usage/memory_usage > 1	CPU usage divided by memory usage greater than 1 (just an example, no special meaning)

Multiple Metric Alert With Labels

1. Current session usage alert, using current session count and max session count metrics combined. Assuming metric names are: current_session, max_session, with one label: database type (database_type), here are some alert examples:

Alert Expression	Alert Condition
current_session/max_session > 0.7	Session usage greater than 70%
current_session{database_type == 'se'}/max_session{database_type == 'se'} > 0.7	SE type database session usage greater than 70%

Q&A

TODO Common Problem Solutions Guide

glossary

references