

产品简介

YTC(yashan trace collector) | 崖山一键收集工具

YTC(崖山一键收集工具)，是一款针对崖山数据库信息与日志收集的工具。

用户定位

- 崖山DBA

产品定位

- 轻量的独立工具
- 开箱即用

场景建议

- 数据库出现故障时
- 数据库出现性能问题时
- 其他任何想要快速收集相关信息时

核心功能

基础信息的收集

- 崖山数据库基本信息(版本号/commit号等)
- 所在服务器的基本信息(操作系统/硬件配置/防火墙等)
- 所在服务器的负载情况(网络流量/CPU占用/I/O负载/内存/磁盘容量等)
- ...

故障信息的收集

- 数据库状态及进程检查
- ADR日志
- coredump文件
- 数据库日志(run.log/alert.log等)
- 操作系统日志(Dmesg/message_log等)
- ...

性能数据的收集

- AWR报告
- 慢日志
- ...

灵活可配的收集策略

- 支持自定义时间周期，绝对灵活的时间周期选择
- 支持自定义收集模块，三大模块均可灵活搭配选择
- 支持自定义路径数据收集，目录或文件均可批量选择
- ...

丰富健全的数据管理

- 支持自定义收集数据的存放路径
- 多种收集数据展示形式(txt/md/html)
- ...

规格说明

产品形态

一个独立的命令行工具

平台支持

- Linux(ubuntu/centos/kylin)

硬件架构

- x86
- arm

部署方式

- 无需部署，开箱即用

产品规格

1.仅支持本地信息收集

2.仅支持针对YashanDB SE的信息收集

参数规格

针对collect子命令的参数有如下约束可供参考：

- t 必须是base , diag , perf三个中的多个，以逗号分割
- r参数的优先级高于-s/-e
- r最大为30d，最小为1m，可以通过配置文件({ytc_home}/config/strategy.toml)修改
- s/-e 格式为yyyy-MM-dd:ss-mm，间隔最大为7d，最小为1m，可以通过配置文件({ytc_home}/config/strategy.toml)修改

功能规格

- 工具会根据指定的时间周期尝试收集服务器对应时段(默认24h内)的历史负载情况相关数据，需要依赖sar(System Activity Reporter)这个常用的系统性能监控工具，建议预装该工具否则历史负载数据无法收集，有效数据是基于sar工具运行起来之后才能被跟踪，且默认只保存30天内的数据。

快速开始

工具获取

1.登录数据库所在的服务器，访问地址获取工具包

```
http://192.168.19.121:9988/product/ycm/release/ycl/
```

2.解压至对应目录(示例默认解压至当前目录)：

```
tar -zxvf yashan-trace-collector-v0.1.0-linux-x86_64.tar.gz
```

3.进入 `yashan-trace-collector` 文件夹执行 `./ytctrl -v` 成功即可

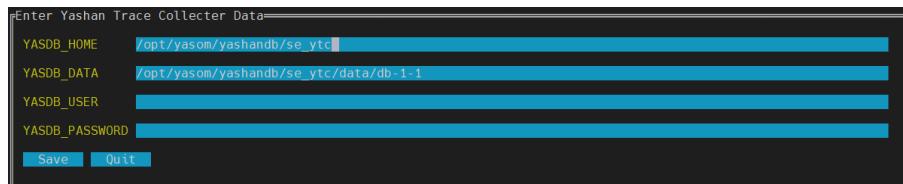
```
-bash-4.2$ ./ytctrl -v
v0.1.0
```

开始收集

强烈建议使用需要被收集的数据库进程所属用户运行该工具且给予该用户sudo权限，或使用root用户运行工具。

执行 `./ytctrl collect` 可以直接开始一次收集动作，所有参数不指定时，工具会默认收集24小时以内的基础信息和故障信息(模块内包含的具体指标可移步[参考手册](#))。

1.进入前置交互页填写相关信息后Save，进入下一步：



- YASDB_HOME：默认填充当前环境变量YASDB_HOME的值，若无，则会尝试匹配当前环境存在的yashandb进程对应的YASDB_HOME路径填充，用户可自行修改并确认
- YASDB_DATA：默认填充当前环境变量YASDB_DATA的值，若无，则会尝试匹配当前环境存在的yashandb进程对应的YASDB_DATA路径填充，用户可自行修改并确认
- YASDB_USER：需要用户提供可访问当前数据库的用户名，至少具备CREATE_SESSION权限，用户可自行修改并确认
- YASDB_PASSWORD：需要用户提供数据库YASDB_USER对应的合法密码，用户可自行修改并确认

2.工具会根据用户提供的信息检查预期收集的数据是否能被正确收集，并展示风险点及相关建议，如下图：

TYPE	COLLECT ITEM	DESCRIPTION	TIPS	COLLECTED?
BASEINFO	YashanDB-Parameter	login yasdb failed: YAS-02143:invalid username/password or user doesn't exist	default to collect parameter from /home/smile/yashandb.ini/h/data/db-1-1/config/yasdb.ini	true
DIAGNOSIS	YashanDB-InstanceStatus	login yasdb failed: YAS-02143:invalid username/password or user doesn't exist	you can enter a correct username and password	false
	YashanDB-ADR	login yasdb failed: YAS-02143:invalid username/password or user doesn't exist	default collect adr from /home/smile/yashandb/data/d-1-1/addr	true
	YashanDB-RunLog	login yasdb failed: YAS-02143:invalid username/password or user doesn't exist	you can enter a correct username and password	false
	Host-SystemLog	current user: smile stat: /var/log/messages permission denied		false
	YashanDB-DatabaseStatus	login yasdb failed: YAS-02143:invalid username/password or user doesn't exist	you can enter a correct username and password	false

Are you want continue collect [y/n] ?

比较常见的风险提示例如：

- 1.you can run with sudo，是因为部分日志信息的收集需要用户具备root权限，若当前执行的用户具备sudo权限，我们建议您使用sudo运行该工具以便收集到更多的信息。
- 2.login yasdb failed，是因为提供的数据库用户名和密码登陆失败，若无法连接数据库则会有较多数据库相关的信息无法准确收集，我们建议尽量提供可正常访问数据库的用户名及密码以便收集到更多的信息。
- 3.current user xxx stat /yyy/zzz permission denied，是因为当前用户没有权限执行需要收集的某项指标对应的文件所在路径的文件，若无-r-x权限则无法收集该部分信息，我们建议您使用满足预期信息数据能被收集到的用户(推荐使用：数据库的启动用户且赋予sudo权限)运行该工具以便收集到

更多的信息。

3.若想继续收集则输入y，若想按照建议操作则选n退出重新执行，如下图(示例继续执行)：

```
Are you want continue collect [y/n] ?
y
The following modules are about to be collected:
baseinfo      | diagnosis      |
-----+-----+
YashanDB-Paramter | YashanDB-ADR      |
-----+-----+
Host-MemoryUsage | YashanDB-RunLog      |
-----+-----+
Host-CPU        | YashanDB-AlertLog      |
-----+-----+
Host-Disk        | YashanDB-ProcessStatus      |
-----+-----+
Host-Network      |                         |
-----+-----+
Host-Memory       |                         |
-----+-----+
Host-NetworkIO     |                         |
-----+-----+
YashanDB-Version   |                         |
-----+-----+
Host-OSInfo        |                         |
-----+-----+
Host-Firewalld     |                         |
-----+-----+
Host-CPUUsage       |                         |
-----+-----+
Host-DiskIO         |                         |
-----+-----+
```

开始收集前会列出本次能够收集的各项指标数据(具体指标可移步[参考手册](#))

4.观察工具收集的具体进度，等待收集完成，如下图：

```
base 100 % [=====] done
YashanDB-Version has been completed
Host-CPUUsage has been completed
Host-DiskIO has been completed
YashanDB-Paramter has been completed
Host-Network has been completed
Host-CPU has been completed
Host-NetworkIO has been completed
Host-Disk has been completed
Host-Memory has been completed
Host-Network has been completed
Host-Firewalld has been completed
Host-CPUUsage has been completed
Host-DiskIO has been completed
Host-Firewalld has been completed
diag 100 % [=====] done
YashanDB-RunLog has failed, err: open /opt/yasom/yashandb/se_ytc/data/db-1-1/diag/metadata/hm_finding for writing into tarball: open /opt/yasom/yashandb/se_ytc/data/db-1-1/diag/metadata/hm_finding for writing into tarball: open /opt/yasom/yashandb/se_ytc/data/db-1-1/diag/metadata/hm_run: permission denied
YashanDB-RunLog has been failed, err: open /opt/yasom/yashandb/se_ytc/data/db-1-1/log/run.log: permission denied
YashanDB-AlertLog has been failed, err: open /opt/yasom/yashandb/se_ytc/data/db-1-1/log/alert/alert.log: permission denied
YashanDB-ProcessStatus has been completed
result was saved to /home/lfx/totest/results/ytic-20230816154927.tar.gz
```

收集过程会动态更新进度，完成后会提示用户结果的存放路径，本次结果存放在 `/home/lfx/totest/results/ytic-20230816154927.tar.gz`，默认收集结果会打包存放在工具执行目录的results路径下，用户也可以使用-o参数自定义路径。

信息查看

1.将收集到的信息包解压，会生成一个同名目录，如下：

```
// 解压文件
-bash-4.2$ tar -zvxf ytc-20230816154927.tar.gz
ytic-20230816154927/
ytic-20230816154927/diag/
ytic-20230816154927/diag/coredump/
ytic-20230816154927/diag/adr/
ytic-20230816154927/diag/adr/hm/
ytic-20230816154927/diag/adr/metadata/
ytic-20230816154927/diag/adr/metadata/hm_finding
ytic-20230816154927/diag/adr/metadata/hm_run
ytic-20230816154927/diag/adr/metadata/incident
ytic-20230816154927/diag/adr/metadata/problem
ytic-20230816154927/diag/adr/trace/
ytic-20230816154927/diag/log/
ytic-20230816154927/diag/log/yasdb/
ytic-20230816154927/diag/log/yasdb/alert.log
ytic-20230816154927/diag/log/yasdb/run.log
ytic-20230816154927/diag/log/system/
```

```

ytic-20230816154927/diag/log/system/dmesg.log
ytic-20230816154927/data/
ytic-20230816154927/data/data-20230816154927.json
ytic-20230816154927/report_static/
ytic-20230816154927/report_static/css/
ytic-20230816154927/report_static/css/morris.css
ytic-20230816154927/report_static/js/
ytic-20230816154927/report_static/js/morris.js
ytic-20230816154927/report_static/js/raphael.min.js
ytic-20230816154927/report-20230816154927.txt
ytic-20230816154927/report-20230816154927.md
ytic-20230816154927/report-20230816154927.html

// 查看当前目录情况
-bash-4.2$ ll
total 96
-rw-r--r--. 1 lfx wheel 32533 Aug 16 15:48 ytic-20230816154835.tar.gz
-rw-r--r--. 1 lfx wheel 32246 Aug 16 15:49 ytic-20230816154855.tar.gz
drwxr-xr-x. 5 lfx wheel 75 Aug 16 15:49 ytic-20230816154927
-rw-r--r--. 1 lfx wheel 32300 Aug 16 15:49 ytic-20230816154927.tar.gz

// 进入ytic-20230816154927目录
-bash-4.2$ cd ytic-20230816154927/
-bash-4.2$ ll
total 620
drwxr-xr-x. 2 lfx wheel 38 Aug 25 14:43 data
drwxr-xr-x. 4 lfx wheel 73 Aug 25 14:43 diag
-rw-r--r--. 1 lfx wheel 264075 Aug 25 14:43 report-20230825144316.html
-rw-r--r--. 1 lfx wheel 96210 Aug 25 14:43 report-20230825144316.md
-rw-r--r--. 1 lfx wheel 266793 Aug 25 14:43 report-20230825144316.txt
drwxr-xr-x. 4 lfx wheel 27 Aug 25 14:43 report_static

```

- data : 本次收集的源数据存放路径，默认会有一份json文件里面包含本次所有收集到的底层数据也是生成report的数据源
- diag : 本次收集的 故障信息 模块相关的源文件存放路径(若无则空)
- report-{生成时间}.html : 本次收集的整体报告文件的html格式文件
- report-{生成时间}.md : 本次收集的整体报告文件的md格式文件
- report-{生成时间}.txt : 本次收集的整体报告文件的txt格式文件
- report_static : html报告依赖的js、css等文件

更多信息可参看[报告解读](#)

最佳实践

该部分为工具使用场景的最小单元拆解指导，用户可以根据实际所需场景自行整合场景单元，形成最佳实践。

- [默认收集](#)
- [指定时间范围](#)
- [指定模块](#)

参数解释

ytic目前支持一个子命令，即 `collect`，以下是针对collect命令的参数解释。

```
-bash-4.2$ ./ytccctl collect -h
Usage: ytccctl collect

The collect command is used to gather trace data.

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

  -t, --type="base,diag"      The type of collection, choose many of (base|diag|perf) and split with ','.
  -r, --range=STRING        The time range of the collection, such as '1M', '1d', '1h', '1m'. If <range> is given,
<start> and <end> will be discard.
  -s, --start=STRING        The start datetime of the collection, such as 'yyyy-MM-dd', 'yyyy-MM-dd-hh', 'yyyy-MM-
dd-hh-mm'.
  -e, --end=STRING          The end timestamp of the collection, 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 collection.
  --include=STRING          Files or directories that need to be additionally collected, it is absolute path and
split with ',', such as '/tmp' or '/tmp,/root,/example.txt'.
  --exclude=STRING          Files or directories that no need to be additionally collected, it is absolute path and
split with ',', such as '/tmp' or '/tmp,/root,/example.txt'.
```

- `-h`: 查看collect子命令的帮助信息
- `-v`: 查看ytic工具的当前版本
- `-c`: 指定工具所使用的配置文件，默认为 `{ytic_home}/config/ytic.toml`
- `-t`: 指定预期收集的模块，总共有3个模块(基础信息base/故障信息diag/性能数据perf)，默认收集base和diag，指定多个时用英文逗号隔开即可
- `-r`: 指定时间周期，默认收集从当前时间往前推24h的时间周期内数据，该配置项全局生效且优先级高于start/end参数，规格约束最大为30d，最小为1m，可以通过配置文件(`{ytic_home}/config/strategy.toml`)修改
- `-s`: 指定收集数据的开始时间点，默认为当前时间往前推24h的时间点，该配置项全局生效且优先级低于-r参数，规格约束格式 `yyyy-MM-dd-ss-mm` 且与end生效形成的时间周期最大为7d，最小为1m，可以通过配置文件(`{ytic_home}/config/strategy.toml`)修改
- `-e`: 指定收集数据的结束时间点，默认为当前时间点，该配置项全局生效且优先级低于-r参数，规格约束格式 `yyyy-MM-dd-ss-mm`，且与start生效形成的时间周期最大为30d，最小为1m，可以通过配置文件(`{ytic_home}/config/strategy.toml`)修改
- `-o`: 指定收集后的数据存放的路径，该路径至少需要具备执行工具当前用户的写权限
- `--include`: 指定某些目录或文件随本次一并收集，多个文件或目录用英文逗号隔开即可
- `--exclude`: 指定额外收集文件时排除的文件，作用于include参数，exclude参数指定的文件将不会被收集，多个文件或目录用英文逗号隔开即可

默认收集

直接执行 `ytctl collect` 即可

默认收集即不指定任何参数，各参数按照默认值执行，具体默认值可参看[参数解释](#)

默认收集具体执行操作指导可参看[快速开始](#)

指定模块

工具获取

1.登录数据库所在的服务器，访问地址获取工具包

```
http://192.168.19.121:9988/product/ycm/release/ytc/
```

2.解压至对应目录(示例默认解压至当前目录)：

```
tar -zxvf yashan-trace-collector-v0.1.0-linux-x86_64.tar.gz
```

3.进入 `yashan-trace-collector` 文件夹执行 `./ytcctl -v` 成功即可

```
-bash-4.2$ ./ytcctl -v  
v0.1.0
```

开始收集

通过-t参数指定

- t: 支持指定数据收集的模块，目前内置3种模块分别是base/diag/perf即基础信息/故障信息/性能数据(模块包含的具体指标数据可查看[参考手册](#))，工具默认会收集base和diag模块。

例如，执行 `./ytcctl collect -t base` 即表示仅收集基础信息模块数据。执行 `./ytcctl collect -t base,diag,perf` 即表示收集基础信息，故障信息和性能数据模块数据。

信息查看

找到工具成功后提示的报告存放路径，即可查看本次收集数据的详细信息(具体报告解读可参看[报告解读](#))。

指定时间范围

工具获取

1.登录数据库所在的服务器，访问地址获取工具包

```
http://192.168.19.121:9988/product/ycm/release/ytc/
```

2.解压至对应目录(示例默认解压至当前目录)：

```
tar -zxvf yashan-trace-collector-v0.1.0-linux-x86_64.tar.gz
```

3.进入 `yashan-trace-collector` 文件夹执行 `./ytcctl -v` 成功即可

```
-bash-4.2$ ./ytcctl -v
v0.1.0
```

开始收集

以下为两种指定时间范围的方案，`-r`的方案优先级高于`-s/-e`即，若指定了`-r`参数，则`-s/-e`本次不会生效。

通过`-r`参数指定

- 使用示例：`-r 1d` 即一天内，`-r 1h` 即一小时内，`-r 1m` 即1分钟内
- `-r`最大为30d，最小为1m，可以通过配置文件(`{ytc_home}/config/strategy.toml`)修改

例如，执行 `./ytcctl collect -r 7h` 即指定收集当前时刻往前推7个小时的数据信息

通过`-s/-e`参数指定

- 使用示例：`-s 2023-06-01 -e 2023-07-01` 即表示收集从6月1号至7月1号的数据信息
- `-s/-e` 格式为`yyyy-MM-dd:ss-mm`，间隔最大为7d，最小为1m，可以通过配置文件(`{ytc_home}/config/strategy.toml`)修改
- `-s`和`-e`可以单独致指定，未指定的参数按照默认值即`-s`为当前时间往前推24h的时刻，`-e`为当前时刻

例如，执行 `./ytcctl collect -s 2023-06-01` 即指定开始时间为23年的6月1号，而结束时间则为默认的当前时刻，但需保证时间周期长度不能超过最大规格30d，若需调整可自行修改对应配置文件(`{ytc_home}/config/strategy.toml`)

信息查看

找到工具成功后提示的报告存放路径，即可查看本次收集数据的详细信息(具体报告解读可参看[报告解读](#))。

指定报告存放路径

工具获取

1. 登录数据库所在的服务器，访问地址获取工具包

```
http://192.168.19.121:9988/product/ycm/release/ytc/
```

2. 解压至对应目录(示例默认解压至当前目录)：

```
tar -zxvf yashan-trace-collector-v0.1.0-linux-x86_64.tar.gz
```

3. 进入 `yashan-trace-collector` 文件夹执行 `./ytctl -v` 成功即可

```
-bash-4.2$ ./ytctl -v  
v0.1.0
```

开始收集

通过-o参数指定

- `-o`: 支持指定收集完成后的文件存放路径，若路径不存在工具会默认创建，若当前执行用户权限不足则无法继续运行

例如，执行 `./ytctl collect -o /home/lfx` 即表示本次收集后的数据包将会存放在 `/home/lfx` 的路径。

使用默认路径

- 不使用`-o`参数指定收集完成后的文件存放路径时，将会使用配置文件中的`output`配置作为默认存储路径，默认保存在 `yashan-trace-collector` 目录下的 `results` 文件夹中

信息查看

找到工具成功后提示的报告存放路径，即可查看本次收集数据的详细信息(具体报告解读可参看[报告解读](#))。

指定收集文件

工具获取

1.登录数据库所在的服务器，访问地址获取工具包

```
http://192.168.19.121:9988/product/ycm/release/ytc/
```

2.解压至对应目录(示例默认解压至当前目录)：

```
tar -zxvf yashan-trace-collector-v0.1.0-linux-x86_64.tar.gz
```

3.进入 `yashan-trace-collector` 文件夹执行 `./ytcctl -v` 成功即可

```
-bash-4.2$ ./ytcctl -v
v0.1.0
```

开始收集

通过--include与--exclude参数灵活指定

- include: 支持自定义新增收集指定路径下的目录或文件数据
- exclude: 该参数基于 --include 使用，支持排除 --include 指定范围内的对应目录或文件数据

1.例如，执行 `./ytcctl collect --include /home/lfx` 即表示本次收集后除了收集默认模块的数据文件以外，还会将 `/home/lfx` 路径下的文件一并打包收集。2.若需要排除掉 --include 指定的路径中某些文件或目录，可以使用 --exclude 参数排除，例如，执行 `./ytcctl collect --include /home/lfx --exclude /home/lfx/test` 即表示收集 `/home/lfx` 路径下除test目录的所有文件数据。

信息查看

找到工具成功后提示的报告存放路径，即可查看本次收集数据的详细信息，若指定了文件则会保存在extra目录即额外数据模块(具体报告解读可参看[报告解读](#))。

参考手册

该部分主要用来详细阐述工具内置的各项数据收集的具体实现以及收集完成后的数据包解释及工具配置项指导，以便用户更好的理解和分析收集到的数据信息及调控工具能力。

- [工具结构](#)
- [基础信息](#)
- [故障信息](#)
- [性能数据](#)
- [报告解读](#)
- [配置文件](#)

若指定了 `--include` 参数则会基于Extra-FileCollect指标项跟踪收集并存放在对应的extra目录。

基础信息

指标名	含义	底层实现	备注
YashanDB-Version	YashanDB版本和commit号	{YASDB_HOME}/bin/yasdb -V	
YashanDB-Parameter	YashanDB服务端配置参数	1. cat{YASDB_DATA}/config/yasdb.ini 2. select * from v\$parameter where value is not null	该项数据为ini文件的内容及v\$parameter视图中不为空的所有参数当前值
Host-OSInfo	服务器操作系统基本信息	github.com/shirou/gopsutil/host的Info()	调用的是golang中的第三方依赖，该依赖底层调用的还是各OS的基础命令
Host-Firewalld	服务器防火墙状态	systemctl is-active firewalld	
Host-CPU	服务器CPU基本信息	github.com/shirou/gopsutil/cpu的Info()	调用的是golang中的第三方依赖，该依赖底层调用的还是各OS的基础命令
Host-Disk	服务器磁盘基本信息	github.com/shirou/gopsutil/disk的Partitions()	调用的是golang中的第三方依赖，该依赖底层调用的还是各OS的基础命令
Host-Network	服务器网卡基本信息	github.com/shirou/gopsutil/net的Interfaces()	调用的是golang中的第三方依赖，该依赖底层调用的还是各OS的基础命令
Host-Memory	服务器内存基本信息	github.com/shirou/gopsutil/mem的VirtualMemory()	调用的是golang中的第三方依赖，该依赖底层调用的还是各OS的基础命令
Host-NetworkIO	服务器网络负载情况	<p>1.历史负载(依赖sar工具) 查看sar是否存在自定义路径配置 sar_dir : /etc/sysconfig/sysstat 若不存在则使用各os默认的路径执行(ubuntu: /var/log/sysstat ; 其他: /var/log/sa) 执行具体查询命令： sar -n DEV -f {sar_dir}/{对应日期的sa文件} -s 03:41:57 -e 15:41:57</p> <p>2.当前负载 若sar存在，则通过sar查询当前网络负载情况，例如执行 sar -n DEV 1 10 每隔1s查询一次总共查询10次 若sar不存在，则通过gopsutil的net.INCounters()查询当前网络负载情况，每隔1s查询一次总共查询10次后计算得出近似的负载情况</p>	<p>1.当sar未安装时无法获取历史负载数据，仅能算出当前负载数据 2.gopsutil是由程序自定义的算法计算得出会存在略微的偏差，仅供参考 3.间隔次数及时长均可通过配置文件调整</p>
Host-CPUUsage	服务器CPU负载情况	<p>1.历史负载(依赖sar工具) 查看sar是否存在自定义路径配置 sar_dir : /etc/sysconfig/sysstat 若不存在则使用各os默认的路径执行(ubuntu: /var/log/sysstat ; 其他: /var/log/sa) 执行具体查询命令： sar -u -f {sar_dir}/{对应日期的sa文件} -s 03:41:57 -e 15:41:57</p> <p>2.当前负载 若sar存在，则通过sar查询当前网络负载情况，例如执行 sar -u 1 10 每隔1s查询一次总共查询10次 若sar不存在，则通过gopsutil的net.INCounters()查询当前网络负载情况，每隔1s查询一次总共查询10次后计算得出近似的负载情况</p>	<p>1.当sar未安装时无法获取历史负载数据，仅能算出当前负载数据 2.gopsutil是由程序自定义的算法计算得出会存在略微的偏差，仅供参考 3.间隔次数及时长均可通过配置文件调整</p>

指标名	含义	底层实现	备注
Host-DiskIO	服务器磁盘I/O负载情况	<p>1.历史负载(依赖sar工具) 查看sar是否存在自定义路径配置 <code>sar_dir : /etc/sysconfig/sysstat</code> 若不存在则使用各os默认的路径执行(ubuntu: <code>/var/log/sysstat</code> ; 其他: <code>/var/log/sa</code>) 执行具体查询命令：<code>sar -d -f {sar_dir}/{对应日期的sa文件} -s 03:41:57 -e 15:41:57</code></p> <p>2.当前负载 若sar存在，则通过sar查询当前网络负载情况，例如执行 <code>sar -d 1 10</code> 每隔1s查询一次总共查询10次 若sar不存在，则通过gopsutil的net.INCounters()查询当前网络负载情况，每隔1s查询一次总共查询10次后计算得出近似的负载情况</p>	<p>1.当sar未安装时无法获取历史负载数据，仅能算出当前负载数据 2.gopsutil是由程序自定义的算法计算得出会存在略微的偏差，仅供参考 3.间隔次数及时长均可通过配置文件调整</p>
Host-MemoryUsage	服务器内存负载情况	<p>1.历史负载(依赖sar工具) 查看sar是否存在自定义路径配置 <code>sar_dir : /etc/sysconfig/sysstat</code> 若不存在则使用各os默认的路径执行(ubuntu: <code>/var/log/sysstat</code> ; 其他: <code>/var/log/sa</code>) 执行具体查询命令：<code>sar -r -f {sar_dir}/{对应日期的sa文件} -s 03:41:57 -e 15:41:57</code></p> <p>2.当前负载 若sar存在，则通过sar查询当前网络负载情况，例如执行 <code>sar -r 1 10</code> 每隔1s查询一次总共查询10次 若sar不存在，则通过gopsutil的net.INCounters()查询当前网络负载情况，每隔1s查询一次总共查询10次后计算得出近似的负载情况</p>	<p>1.当sar未安装时无法获取历史负载数据，仅能算出当前负载数据 2.gopsutil是由程序自定义的算法计算得出会存在略微的偏差，仅供参考 3.间隔次数及时长均可通过配置文件调整</p>

故障信息

指标名	含义	底层实现	备注
YashanDB-ProcessStatus	数据库进程信息	data路径匹配进程	
YashanDB-InstanceStatus	数据库实例状态	<pre>select status from v\$instance;</pre>	依赖提供的数据库用户及密码能够正确连接数据库并可执行对应sql查询语句
YashanDB-DatabaseStatus	数据库状态	<pre>select status,open_mode as openMode from v\$database</pre>	1.该项数据依赖数据库实例状态为OPEN且用户具备相关权限时才能正常查询 2.主要查询数据库的运行状态与开启模式
YashanDB-ADR	数据库的自动诊断数据	<pre>select name,value from v\$parameter where name=DIAGNOSTIC_DEST</pre> 找到诊断数据的主路径	1.若任何异常场景导致无法从数据库中获取该参数对应的ADR主路径，工具会默认在YASDB_DATA路径下寻找diag目录 2.需要执行工具的用户具备对应文件的执行权限
YashanDB-RunLog	数据库运行日志	<pre>select name,value from v\$parameter where name=RUN_LOG_FILE_PATH</pre> 找到运行日志的主路径	1.若任何异常场景导致无法从数据库中获取该参数对应的主路径，工具会默认在YASDB_DATA路径下寻找log目录下的run.log 2.需要执行工具的用户具备对应文件的执行权限
YashanDB-AlertLog	数据库运行日志	基于数据库的DATA路径找到log目录下的alert.log	需要执行工具的用户具备对应文件的执行权限
YashanDB-Coredump	服务器上的coredump文件	1.查看文件 /proc/sys/kernel/core_pattern 配置的coredump文件生成路径 2.重定向程序 (abrt-hook-ccpp、systemd-coredump) to /var/spool/abrt、绝对路径、相对路径to home/bin下寻找core	
Host-KernelLog	内核日志	dmesg命令筛选	主要是Dmesg数据
Host-SystemLog	操作系统日志	/var/log/messages 或者是 /var/log/syslog	

性能数据

指标名	含义	底层实现	备注
YashanDB-AWR	数据库AWR报告	基于工具指定的时间范围内的快照执行命令生成一次AWR报告 <code>exec sys.dbms_awr.awr_report({db_id},{instance_NUMBER}, snap_start_id), {snap_end_id};</code> , 再执行yasql命令	<ul style="list-style-type: none"> 1.依赖数据库可以正常连接，且连接数据的用户必须为sys用户 2.若由于快照过大或者快照间差异过大而导致的AWR报告生成耗时太长，本次收集可能会失败，可以通过配置文件调整最长耗时(默认10min) 3.若工具指定的收集时段内快照数量未达到2个则AWR默认无法成功收集 4.若工具指定的收集时段内数据库出现过重启操作则本次AWR报告无法成功收集
YashanDB-SlowSQL	数据库慢sql信息	通过数据库慢sql相关的配置参数 <code>ENABLE_SLOW_LOG</code> 、 <code>SLOW_LOG_TIME_THRESHOLD</code> 、 <code>SLOW_LOG_SQL_MAX_LEN</code> 、 <code>SLOW_LOG_OUTPUT</code> 查询对应文件的慢sql记录，并且查询视图 <code>SLOW_LOG\$</code> 中对应时间段内的数据	查询视图依赖数据库用户正常访问数据库且具备对应查询权限，若配置了存在slow.log文件则要求运行工具的用户具备该文件的执行权限

报告解读

报告获取

执行完ytc工具后根据工具指引到达对应路径，进入对应路径查看当前文件列表，可以看到各压缩包有对应的生成时间，解压想要查看的包。

这里我们以数据最全的场景作为讲解示例

进入报告存放的文件夹，文件夹内存放收集结果的压缩包，压缩包命名规则为 `ytc-收集时间.tar.gz`，根据收集时间找到对应的压缩包：

```
-bash-4.2$ cd results
-bash-4.2$ ll
total 344K
-rw-r--r--. 1 lfx wheel 341K Aug 28 15:27 ytc-20230828152715.tar.gz
```

解压对应的收集结果压缩包：

```
-bash-4.2$ tar -zxvf ytc-20230828152715.tar.gz
```

进入解压后的文件夹，文件夹中存放有本次收集文件，不同格式的报告，报告的原始json数据和html报告的css样式等：

```
-bash-4.2$ cd ytc-20230828152715
-bash-4.2$ ll
total 788K
drwxr-xr-x. 2 lfx wheel 38 Aug 28 15:27 data
drwxr-xr-x. 5 lfx wheel 44 Aug 28 15:27 diag
drwxr-xr-x. 4 lfx wheel 31 Aug 28 15:27 extra
drwxr-xr-x. 4 lfx wheel 32 Aug 28 15:27 perf
-rw-r--r--. 1 lfx wheel 376K Aug 28 15:27 report-20230828152715.html
-rw-r--r--. 1 lfx wheel 100K Aug 28 15:27 report-20230828152715.md
-rw-r--r--. 1 lfx wheel 310K Aug 28 15:27 report-20230828152715.txt
drwxr-xr-x. 4 lfx wheel 27 Aug 28 15:27 report_static
```

各个文件夹和文件对应的内容如下：

data文件夹：存放报告原始的json数据
diag文件夹：存放故障信息收集模块收集到的文件
perf文件夹：存放性能数据收集模块收集到的文件
extra文件夹：存放额外收集模块收集到的文件
report_static文件夹：存放html格式报告所需的css和js文件
report-[收集时间].html：html格式的报告文件
report-[收集时间].md：markdown格式的报告文件
report-[收集时间].txt：txt格式的报告文件

示例收集执行的收集命令：sudo ./ytcctl collect -t base,diag,perf --include /tmp/test,/test

详细解读

report

一次收集会产生三种html、markdown和txt三种格式的报告。以html格式的报告为例对报告内容进行解读，html格式的报告包括两个主要部分：

- 报告概览信息
- 收集项收集信息

报告概览信息

报告概览信息包括基础概览和收集项概览

基础概览

基础概览包含该次收集过程所收集的数据库DATA路径，使用的数据库用户名，收集的时间范围，收集的模块和收集的执行时间等信息，主要内容如下表所示（以本次收集为例）：

概览项	概览值
收集类型	基础信息、诊断信息、性能调优信息、额外收集项
收集范围--起始时间	2023-08-27 15:27
收集范围--截止时间	2023-08-28 15:28
YashanDB信息--YASDB_HOME	/opt/yasom/yashandb/test/yashandb/22.2.4.0
YashanDB信息--YASDB_DATA	/opt/yasom/yashandb/test/data/db-1-1
数据生成用户(用于收集YashanDB信息)	sys
额外的收集文件	/tmp/test
收集结果存放目录	/home/golang/code/yashan-trace-collector/build/yashan-trace-collector/results
任务开始时间	2023-08-28 15:27:15
任务结束时间	2023-08-28 15:27:25

收集项概览

收集项概览信息展示了本次收集过程中各模块所收集的指标名称

基础信息	诊断信息	性能调优信息	额外收集项
数据库版本	数据库进程信息	AWR报告	额外文件收集
数据库配置	数据库实例状态	慢SQL	
操作系统信息	数据库状态		
防火墙配置	CoreDump		
CPU	数据块rnn.log日志		
内存	数据块alert.log日志		
磁盘	数据块ADR日志		
网络配置	操作系统日志		
CPU占用分析	操作系统内核日志		
内存容量检查			
网络流量			
磁盘I/O			

收集项收集信息

收集项收集信息包含本次收集过程中各个收集模块所收集的数据，本次收集过程包含基础信息，诊断信息，性能调优信息和额外收集项（以本次收集为例）

基础信息收集模块

版本信息：记录所收集的数据库版本信息和commit号

版本信息
YashanDB Server Release 22.2.4.0 x86_64 5314095

数据库配置：包含数据库实例配置文件（yasdb.ini）和数据库实例参数视图（v\$parameter）

1.2.1 数据库实例配置文件：yasdb.ini	
参数名称	参数值
CHARACTER_SET	UTF8
CONTROL_FILES	('"/dbfiles/ctrl1", "/dbfiles/ctrl2")
DIN_ADDR	127.0.0.1:1600
LISTEN_ADDR	192.168.4.177:1688
NODE_ID	1-1:1
RUN_LOG_LEVEL	DEBUG
_CLUSTER_ID	c1aabc31e2e1720e06194b49913da4a8

1.2.2 数据库实例参数视图：v\$parameter	
参数名称	参数值
AC_MAX_SOURCE_SLICE_COUNT	20
AC_SLICE_THRESHOLD_SIZE	64M
ARCHIVE_LOCAL_DEST	?/archive
ARCH_CLEAN_IGNORE_MODE	NONE
ARCH_CLEAN_LOWER_THRESHOLD	12G
ARCH_CLEAN_UPPER_THRESHOLD	16G
AUDIT_FLUSH_INTERVAL	100
AUDIT_QUEUE_SIZE	16M
AUDIT_QUEUE_WRITE	TRUE

数据库实例参数视图只展示了视图中值不为空的参数的参数名称和参数值

操作系统信息：当前主机的操作系统类型、开机时间和内核版本等信息

检查项	检查结果
主机名称	mg_4
开机时间	2022-05-26 04:36:51
操作系统	linux
发行版本	centos 7.9.2009 (rhel系列)
内核版本	3.10.0-1160.el7.x86_64
内核架构	x86_64

防火墙配置：当前主机的防火墙状态（开启/关闭）

防火墙状态
已关闭

CPU：当前主机的CPU型号、核心数量、频率，厂商标识ID等基础信息

检查项	检查结果
CPU型号	Intel Xeon Processor (Skylake, IBRS)
CPU核心数量	物理CPU核心: 4, 虚拟CPU核心: 4
CPU主频	@ 2.10GHz
厂商标识ID	GenuineIntel

内存：当前主机的内存相关信息，通常包含系统内存和交换分区的信息

	内存大小	已使用	空间	共享内存	缓冲/缓存	可用
系统内存	15.51G	7.51G	287.32M	135.52M	7.72G	7.54G
交换分区	7.87G	658.46M	7.23G	/	4.04M	

- 内存大小：系统总内存的大小
- 已使用：已使用的内存量，包括正在使用和缓存中的内存
- 空闲：未被使用的内存量，包括空闲内存和用于内核缓存的内存
- 共享内存：共享内存的大小，这部分内存被多个进程共享使用
- 缓冲/缓存：用于内核缓存的内存量。它包括文件系统缓存和其他缓冲区
- 可用：可用的内存量，表示当前可以立即分配给新进程或为现有进程提供的内存。它考虑到了文件系统缓存的部分，因此可以被系统快速分配

磁盘：当前主机各个磁盘的信息，包括设备名称、文件系统类型、磁盘大小、已使用空间、可用空间、使用率、挂载点和挂载选项等信息

磁盘设备	文件系统类型	磁盘大小	已使用	可用	使用率	挂载路径	挂载选项
/dev/dm-0	xfs	49.98G	21.37G	28.6G	42.77%	/	rw,relatime
/dev/vda1	xfs	1014M	182.98M	831.02M	18.05%	/boot	rw,relatime
/dev/dm-2	xfs	141.05G	25.24G	115.81G	17.90%	/home	rw,relatime

- 磁盘设备：磁盘设备名称
- 文件系统类型：当前磁盘的文件系统类型
- 磁盘大小：文件系统总大小
- 已使用：已使用的磁盘空间大小
- 可用：可用的磁盘空间大小
- 挂载路径：文件系统的挂载点
- 挂载选项：挂载文件系统时指定的参数和行为

网络配置：当前主机的网络接口名称、对应的IP地址和MAC地址等信息

网络接口	IP地址	MAC地址
lo	IPv4: 127.0.0.1/8 IPv6: ::1/128	
eth0	IPv4: 192.168.4.177/24 172.25.1.1/16 192.168.4.231/24 IPv6: fe80::5e47:dd83:548:9fd6/64 fe80::976a:a0eb:2e2b:9e7d/64	52:54:00:c7:5b:f5
docker0	IPv4: 172.17.0.1/16 IPv6: fe80::42:35ff:fe80:3ae4/64	02:42:35:B0:3a:e4

CPU占用分析：当前主机的CPU占用情况，包含历史CPU占用情况和当前CPU占用情况



CPU占用信息的详细数据以表格的信息呈现，具体包括：

- 时间：数据的采集时刻
- user：用户空间进程所花费的CPU时间
- nice：以较低优先级运行的用户空间进程所花费的CPU时间
- system：内核空间进程所花费的CPU时间
- iowait：CPU等待磁盘 I/O 完成的时间
- steal：发生CPU窃取的时间
- idle：CPU处于空闲状态的时间

CPU使用率 (user+system) 以时序图的形式呈现，便于直观表示CPU使用率的波动情况。

服务器负载信息（CPU占用情况、内存使用情况、网络IO情况和磁盘IO情况）是基于System Activity Reporter（SAR）工具采集的数据，如果服务器上未安装SAR工具或SAR工具未保留指定时刻的历史监控数据，那么将默认不采集历史负载信息，当前负载信息也将使用自定义算法采集。
历史负载信息收集指定时间范围内的数据，其数据采集的时间间隔受操作系统定时任务cron的配置影响，默认间隔10分钟采集一次数据。
当前负载信息收集当前到未来10秒内（由配置文件控制）的数据。

内存容量检查：当前主机系统内存的使用情况，包含历史内存使用情况和当前内存使用情况

时间	总内存	空闲内存	已使用	使用率	缓冲(buffers)	缓存(cache)	可用	已提交	已提交的内存占总内存的百分比	活跃内存大小	非活跃内存的大小	已修改但尚未写入磁盘的脏页大小	真实内存使用率
2023-08-27 15:40:01	15.51G	348.16M	15.17G	97.81%	0B	7.04G	0B	9.87G	42.20%	8.26G	4.67G	92K	52.45%
2023-08-27 15:50:01	15.51G	236.22M	15.28G	98.51%	0B	7.04G	0B	9.87G	42.20%	8.37G	4.67G	96K	53.15%
2023-08-27 16:00:01	15.51G	338.86M	15.18G	97.87%	0B	7.04G	0B	9.87G	42.20%	8.27G	4.67G	120K	52.50%
2023-08-27 16:10:01	15.51G	334.56M	15.19G	97.99%	0B	7.04G	0B	9.87G	42.20%	8.27G	4.67G	60K	52.51%
2023-08-27 16:20:01	15.51G	228M	15.29G	98.56%	0B	7.04G	0B	9.87G	42.20%	8.38G	4.67G	36K	53.18%
2023-08-27 16:30:01	15.51G	324.24M	15.2G	97.96%	0B	7.04G	0B	9.87G	42.20%	8.28G	4.67G	108K	52.56%
2023-08-27 16:40:01	15.51G	226.23M	15.29G	98.58%	0B	7.04G	0B	9.87G	42.20%	8.38G	4.67G	120K	53.17%
2023-08-27 16:50:01	15.51G	330.01M	15.19G	97.92%	0B	7.04G	0B	9.87G	42.20%	8.28G	4.67G	88K	52.51%
2023-08-27 17:00:01	15.51G	219.56M	15.3G	98.62%	0B	7.05G	0B	9.87G	42.20%	8.38G	4.67G	88K	53.20%

内存使用率

内存容量详细信息以表格形式呈现，具体包括：

- 时间：数据的采集时刻
- 总内存：系统内存的总大小
- 空闲空间：空闲内存的大小
- 已使用：已使用的内存大小，包含了缓冲/缓存的大小
- 使用率：已使用/总内存*100%
- 缓冲：用作缓冲区的内存大小
- 缓存：用作缓存的内存大小
- 已提交：提交的内存（已分配但尚未写入到磁盘）的大小
- 已提交的内存占总内存的百分比：已提交/总内存*100%
- 活跃内存大小：活动内存（正在使用的内存）的大小
- 非活跃内存大小：非活动内存（未使用但仍保留在内存中）的大小
- 已修改但尚未写入磁盘的脏页大小：脏页面（已被修改但尚未写入到磁盘）的大小
- 真实内存使用率：(总内存-空闲内存-缓冲-缓存)/总内存*100%

内存使用率和真实内存使用率以时序图的形式呈现：便于直观展示内存的使用情况。

由于缓冲区和缓存可以复用，在计算内存使用情况，需要考虑缓冲区和缓存。在某些操作系统上（如CentOS），使用率字段只考虑了实际使用的内存，未考虑缓冲区和缓存，因此数据比实际的内存使用率大。

真实内存使用率在计算时考虑到了缓冲和缓存情况，是操作系统真实的内存使用率。

网络流量：当前主机的不同网络接口的网络IO情况

时间	每秒钟接收的数据包数量	每秒钟发送的数据包数量	每秒钟接收的数据量	每秒钟发送的数据量	每秒钟接收的压缩数据包数量	每秒钟发送的压缩数据包数量	每秒钟接收的多播数据包数量
2023-08-27 11:50:01	66.15	56.61	8.07K	6.56K	0	0	0
2023-08-27 12:00:02	66.42	56.74	8.09K	6.57K	0	0	0
2023-08-27 12:10:01	66.9	56.66	8.13K	6.61K	0	0	0
2023-08-27 12:20:01	66.59	56.61	8.1K	6.56K	0	0	0
2023-08-27 12:30:01	66.67	56.53	8.11K	6.55K	0	0	0
2023-08-27 12:40:01	66.25	56.54	8.08K	6.51K	0	0	0
2023-08-27 12:50:01	67	56.75	8.13K	6.87K	0	0	0
2023-08-27 13:00:01	66.84	56.69	8.12K	6.58K	0	0	0
2023-08-27 13:10:01	66.62	56.53	8.1K	6.5K	0	0	0

网络流量

网络流量详细数据以表格形式呈现，具体包括：

- 时间：数据的采集时间
- 每秒钟接收的数据包数量：表示从网络接口接收到的每秒数据包的数量
- 每秒钟发送的数据包数量：表示从网络接口发送出去的每秒数据包的数量
- 每秒钟接收的数据量：表示从网络接口接收到的每秒数据量
- 每秒钟发送的数据量：表示从网络接口发送出去的每秒数据量
- 每秒钟接收的压缩数据包数量：表示从网络接口接收到的每秒压缩数据包的数量

- 每秒钟发送的压缩数据包数量：表示从网络接口发送出去的每秒压缩数据包的数量
- 每秒钟接收的多播数据包数量：表示从网络接口接收到的每秒多播数据包的数量

发送流量和接收流量以时序图的形式呈现，便于直观表示网络的IO情况。

磁盘IO：当前主机的各个磁盘的IO情况



磁盘IO的详细信息以表格的形式呈现，具体包括：

- 时间：数据的采集时刻
- IOPS：每秒钟完成的 I/O 操作数。表示每秒钟磁盘设备处理的读取和写入请求总数
- 每秒钟读取：表示每秒钟从磁盘设备读取的数据量
- 每秒钟写入：表示每秒钟写入到磁盘设备的数据量
- 平均每个请求的扇区数：表示每个I/O请求的平均请求扇区数
- 平均队列长度：表示等待处理的请求队列的平均长度
- 平均I/O请求的等待时间：表示从请求提交到完成所需的平均时间
- 平均I/O请求的服务时间：表示每个 I/O 请求的平均服务时间
- 设备的利用率：表示设备在指定时间内的使用率

磁盘读写和IOPS情况用时序图呈现，便于直观表示磁盘的IO情况

诊断信息

数据库进程信息：数据库进程的PID，命令行，所属用户，创建时间，CPU和内存使用率以及状态信息

进程ID	命令行	所属用户	创建时间	CPU使用率	内存使用率	状态
15591	/opt/yasom/yashandb/test/yashandb/22.2.4.0/bin/yasdb nomount -D /opt/yasom/yashandb/test/data/db-1-1	golang	2023-08-22 18:47:37	2.20%	2.48%	S

数据库实例状态：数据库V\$INSTANCE视图中的STATUS字段

状态	开启模式
OPEN	

数据库状态：数据库V\$DATABASE视图中的STATUS和OPEN_MODE字段

状态	开启模式
NORMAL	READ_WRITE

coredump：记录收集到的coredump文件的路径

存放路径
./diag/coredump

数据库run.log日志：记录收集到的指定时间范围内的数据库run.log路径

存放路径
./diag/log/yasdb/run.log

数据库alert.log：记录收集到的指定时间范围内的数据库alert.log路径

存放路径
./diag/log/yasdb/alert.log

数据库ADR日志：记录收集到的数据库ADR日志文件路径

存放路径
./diag/adr

操作系统日志：记录收集到的指定时间范围内的操作系统日志路径

2.8.1 messages
存放路径
./diag/log/system/messages.log
2.8.2 syslog
存放路径
./diag/log/system/syslog.log

操作系统内核日志：记录收集到的指定时间范围内的操作系统内核日志路径

存放路径
./diag/log/system/dmesg.log

性能调优信息

AWR报告：记录本次收集生成的AWR报告的路径

存放路径
./perf/awr/awrpt-20230827152715-20230828152813.html

慢SQL参数：数据库慢SQL日志相关的参数配置

慢SQL参数名称	参数值
ENABLE_SLOW_LOG	FALSE
SLOW_LOG_OUTPUT	FILE
SLOW_LOG_FILE_PATH	7/log/slow
SLOW_LOG_TIME_THRESHOLD	1000
SLOW_LOG_SQL_MAX_LEN	2000

SLOW_LOG\$系统表：从数据库SLOW_LOG\$系统表收集到的指定时间范围内的慢日志信息

DATABASE_NAME	USER_NAME	START_TIME	USER_HOST	QUERY_TIME	ROWS_SENT	SQL_ID	SQL_TEXT

慢SQL日志文件：收集到的指定时间范围内的数据库慢日志文件路径

存放路径
./perf/slowsql/slow.log

额外收集项

额外文件收集：指定收集文件的原路径和当前路径的映射关系

当前路径	源文件路径
./extra/test	/tmp/test
./extra/_test	/test

存在同名文件或文件夹时会将深度较小的文件或文件夹进行重命名，如同时收集/test和/tmp/test文件夹时，/test文件夹下的内容保存在./extra/_test文件夹下，而/tmp/test文件夹下的内容会保存在./extra/test文件夹下

部分收集模块需要收集文件，报告中记录了所收集文件的保存路径，具体文件存放在对应模块的文件夹下，下面将对收集结果中各文件夹的内容进行解读。

base

base信息由于没有源文件数据需要保存，所以无该模块的文件夹，有效数据均在report文件中可以查阅。

diag

diag文件夹存放故障信息收集模块收集的文件，其文件目录结构如下：

-bash-4.2\$ cd ytc-20230828152715/diag/
-bash-4.2\$ tree
.
├── adr
│ ├── hm
│ ├── metadata
│ ├── hm_finding
│ ├── hm_run
│ ├── incident
│ └── problem
│ └── trace
└── coredump
└── log

```

    └── system
        ├── dmesg.log
        ├── messages.log
        └── syslog.log
    └── yasdb
        ├── alert.log
        └── run.log

```

8 directories, 9 files

- adr文件夹存放数据库的adr信息，对应所收集数据库 `${YASDB_DATA}/diag` 文件夹下的内容
- coredump文件夹存放收集到的core文件
- log文件夹下存放收集到的主机日志和数据库日志，包括主机操作系统内核日志（dmesg.log）、主机messages日志（messages.log）、主机syslog（syslog.log）、数据库run.log如alert.log

perf

perf文件夹存放性能数据模块收集到的文件，其目录结构如下所示：

```

-bash-4.2$ cd ytc-20230828152715/perf/
-bash-4.2$ tree
.
└── awr
    └── awrrpt-20230827152713-20230828152813.html
└── slowsql
    └── slow.log

```

2 directories, 2 files

- awr文件夹存放本次收集生成的数据库AWR报告
- slowsql文件夹存放收集到的数据库慢日志文件

extra

extra文件夹存放本次额外收集的文件

- 如果--include参数指定的多个文件或文件夹存在父子关系，将会合并收集，只保留父文件夹。
- 如果文件夹或文件存在同名的情况，会根据一定的规则将层级较浅的文件夹重命名。
- 收集前后文件路径映射关系可参照报告中的额外文件收集模块。

data

data文件夹存放本次收集的数据

```

-bash-4.2$ cd ytc-20230828152715/data/
-bash-4.2$ tree
.
└── data-20230828152715.json

0 directories, 1 file

```

数据的存储形式json文件，文件命名格式为data-{时间}.json

report_static

report_static文件夹存放html格式报告所需的js和css文件

```

-bash-4.2$ cd ytc-20230828152715/report_static/
-bash-4.2$ tree
.
└── css
    └── morris.css
└── js

```

```
|__ morris.js  
└__ raphael.min.js  
  
2 directories, 3 files
```

配置文件

当前版本工具的有效配置文件路径：`{ytc_home}/config/strategy.toml`

默认出厂配置

```
[collect]
max_duration = "30d"
min_duration = "1m"
network_io_discard = "^lo$,^veth.*|^virbr.*|^br.*|^tap.*|^tun.*|^docker.*|^flannel.*"
output = "./results"
range = "24h"
scrape_interval = 1
scrape_times = 10
awr_timeout = "10m"
```

- max_duration: 工具收集信息的时间范围最大值即指定的-r或-s/-e生效的功能，默认30天
- min_duration : 工具收集信息的时间范围最小值即指定的-r或-s/-e生效的功能，默认1分钟
- network_io_discard : 查询服务器网络负载情况时，默认丢弃的数据
- output : 收集到的数据最后存放的主路径，默认在工具运行当前目录的results目录下
- range:工具收集的时间范围，默认为从当前时间往前24h
- scrape_interval : 网络负载相关数据收集时的时间间隔，默认为1s
- scrape_times : 网络负载相关数据收集时基于时间间隔重复的总次数，默认为10次
- awr_timeout : AWR生成的最大超时时间，默认10分钟

其他非默认配置项

- `sar_dir` : sar的历史监控数据的存放路径，默认为空，工具会根据系统配置获取该路径。极端场景下用户可指定具体路径生效
- `core_dump_path` : core dump文件主路径，默认为空，工具会根据系统配置获取该路径，极端场景下用户可指定具体路径生效
- `core_file_key` : core dump文件的文件名关键字，默认为空，工具会根据系统配置获取关键字，极端场景下用户可指定具体路径生效，建议与 `core_dump_path` 组合使用

以上所有配置项均可按规范自行调整，再次执行时生效

Q&A

TODO 常见问题解决方案指引

术语解释

参考资料