

Malluma V1.0

Performance Analysis User's Guide

目录

1	分析系统启动.....	3
2	用户管理.....	3
3	工程管理.....	4
4	分析选择.....	6
4.1	新建分析.....	6
4.2	导入分析结果.....	6
5	开始分析.....	7
5.1	分析类型选择.....	7
5.1.1	General Exploration 分析.....	7
5.1.2	Scheduling 分析	7
5.1.3	Disk IO 分析	8
5.1.4	锁和等待分析.....	8
5.1.5	LLC&DDR 分析	9
5.1.6	Java Mixed-Mode 分析.....	9
5.1.7	Network Input and Output 分析.....	10
5.2	分析对象选择及参数配置.....	10
5.2.1	Launch Application.....	11
5.2.2	Attach to Process	11
5.2.3	Profile System.....	12
5.2.4	命令行脚本展示.....	13
5.3	分析启动及状态反馈.....	13
5.3.1	启动分析.....	13
5.3.2	状态反馈.....	14
5.4	分析结果查看.....	15
5.4.1	Summary 分析	15
5.4.2	Function 分析.....	17
5.4.3	Timechart 分析	20
5.4.4	Zoomview 分析	21
5.4.5	Code 分析	22
5.4.6	Resource 分析.....	24
5.4.7	Control Flow 分析	25
5.4.8	Latency 分析	26
5.4.9	Disk IO 分析	27
5.4.10	Locks and Waits 分析.....	27
5.4.11	Locks and Waits Code 分析.....	28
5.4.12	LLC&DDR 分析	29
5.4.13	Flame Graph 分析.....	29
5.4.14	Network Input and Output 分析.....	30

6	命令行界面支持.....	31
6.1	功能介绍.....	31
6.2	命令参数.....	33
6.3	示例.....	34
6.3.1	服务器数据采集.....	34
6.3.2	Android 手机终端数据采集.....	35
6.3.3	rtos 单板数据采集	35

1 分析系统启动

完成 Malluma 分析系统安装、配置之后（参考 **Install Guide**），输入 Web 地址栏输入 Server IP 地址启动分析系统。用户登陆后（如图 1.1），自动加载 Sever 中的已有工程及结果到工程导航栏 **Project Navigator**（如图 1.2）。

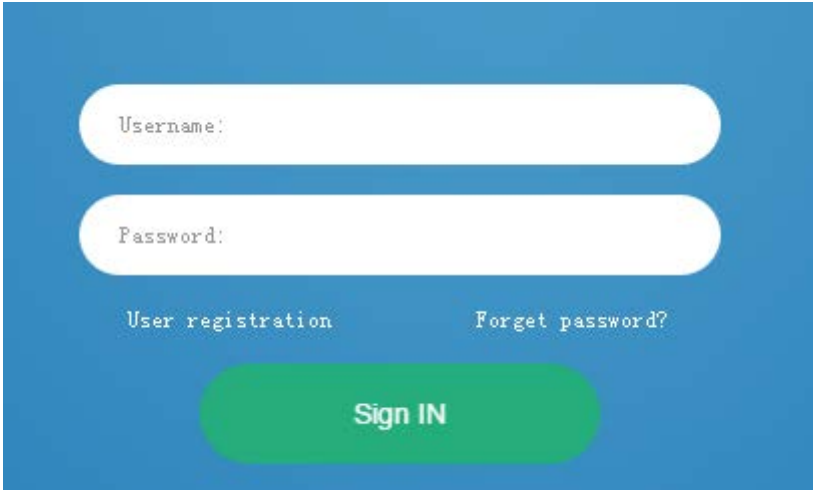


图 1.1

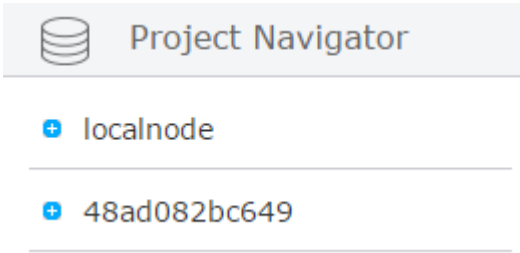


图 1.2

2 用户管理

管理员账号登陆 Malluma 可以进行用户管理，点击“User management”（如图 2.1），弹出对话框添加/删除/更新用户（如图 2.2），可以添加管理员用户或普通用户：

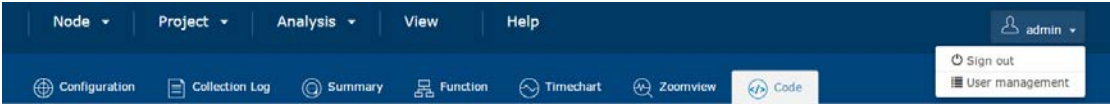


图 2.1

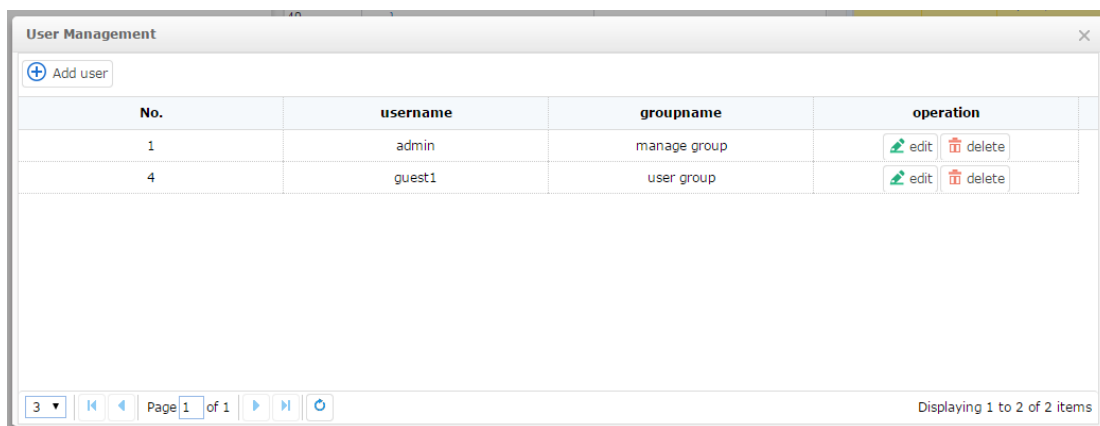


图 2.2

如果是普通用户登录，只支持浏览，无法新建分析。

3 工程管理

创建工程，选择 **Project** 菜单中 **New Project**（如图 3.1），弹出对话框输入工程名（如图 3.2）。

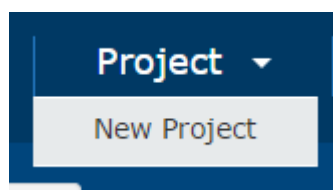


图 3.1

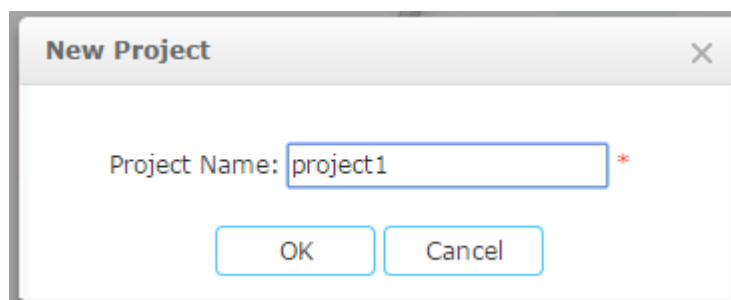


图 3.2

双击选中工程，展开工程下的采集结果，此时可以新建分析。双击已有的采集分析结果，加载分析结果到 **Summary** 等分析界面（如图 3.3）。

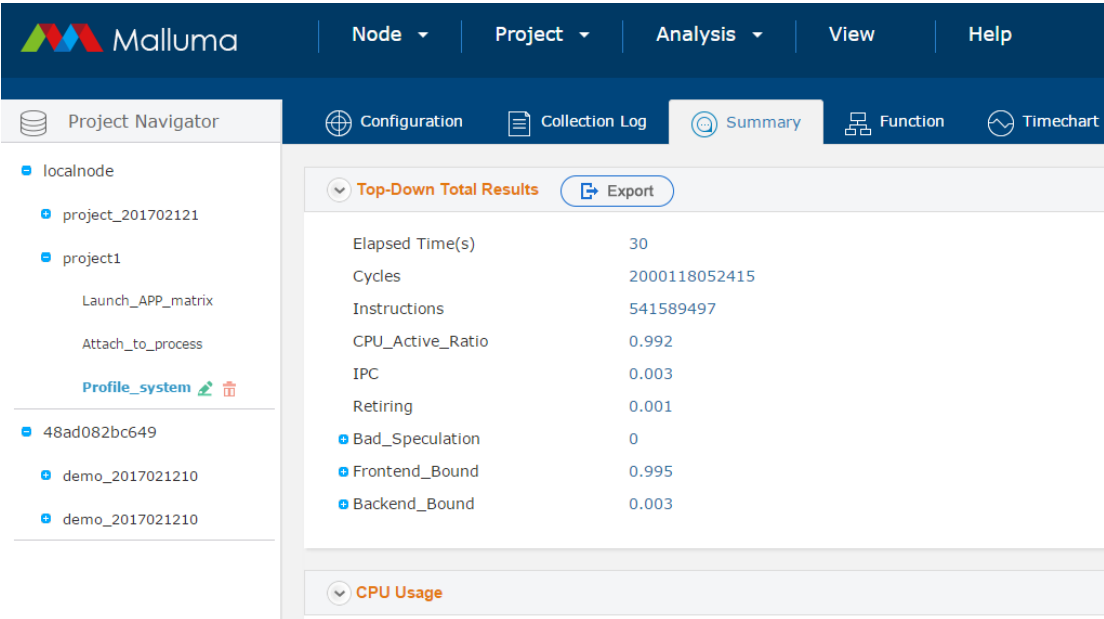


图 3.3

可重命名或删除工程名和结果名（如图 3.4），重命名直接在结果名称上进行编辑（如图 3.5），删除弹出确认对话框（如图 3.6）。

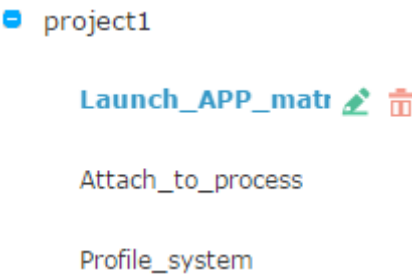


图 3.4

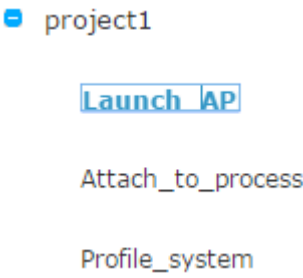


图 3.5

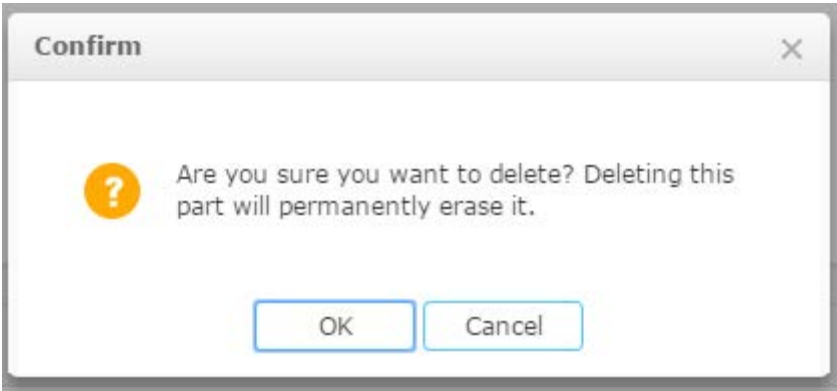


图 3.6

4 分析选择

4.1 新建分析

新建分析结果，选择 Analysis 菜单中 New Analysis（如图 4.1），显示 Configuration Target 分析类型和对象选择页面，可进行相应的设置。注意：先选中任一工程名。

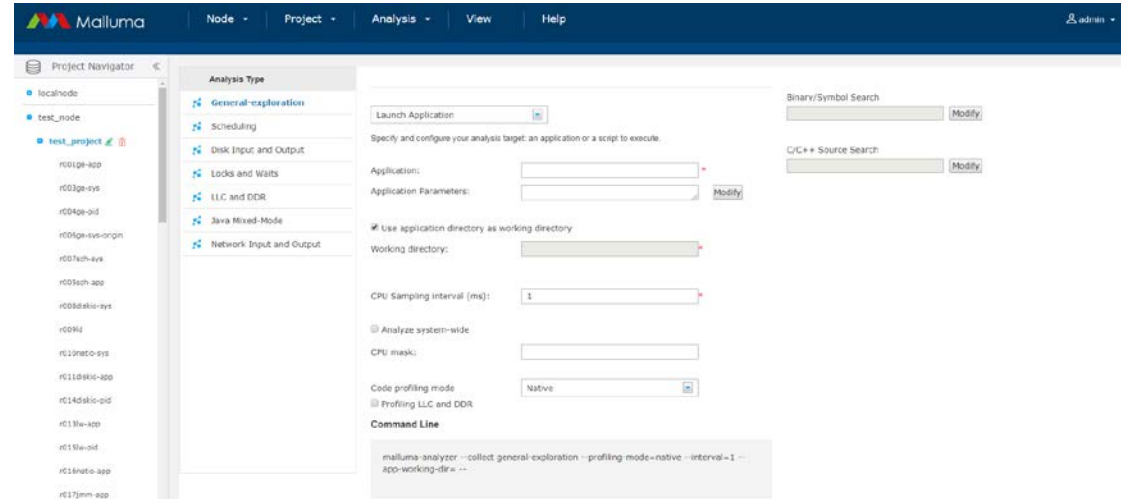


图 4.1

4.2 导入分析结果

导入分析结果，选择 Analysis 菜单中 Import Results 弹出导入对话框（如图 4.2），从后台选择已有的采集结果文件夹加载到指定的工程目录下。

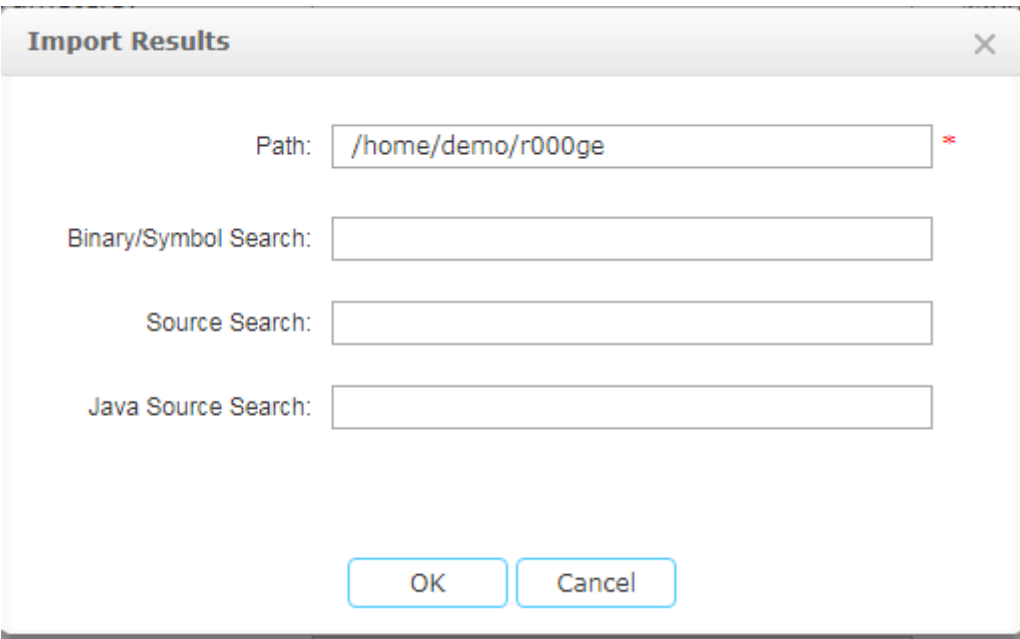


图 4.2

5 开始分析

5.1 分析类型选择

选择分析类型，从哪些维度来分析应用的性能瓶颈，比如微架构级 top-down 模型、算法级锁与等待、平台级 Disk I/O 分析等。

5.1.1 General Exploration 分析

General Exploration 分析，即微架构 top-down 模型分析方法，详见 Tuning Methodology 介绍。可选中分析方法 General-exploration，如图 5.1.1。








Analysis Type	
	General-exploration
	Scheduling
	Disk Input and Output
	Locks and Waits
	LLC and DDR
	Java Mixed-Mode
	Network Input and Output

图 5.1.1

5.1.2 Scheduling 分析

Scheduling 分析，即调度轨迹分析方法，可选中分析方法 Scheduling，如图 5.1.2。

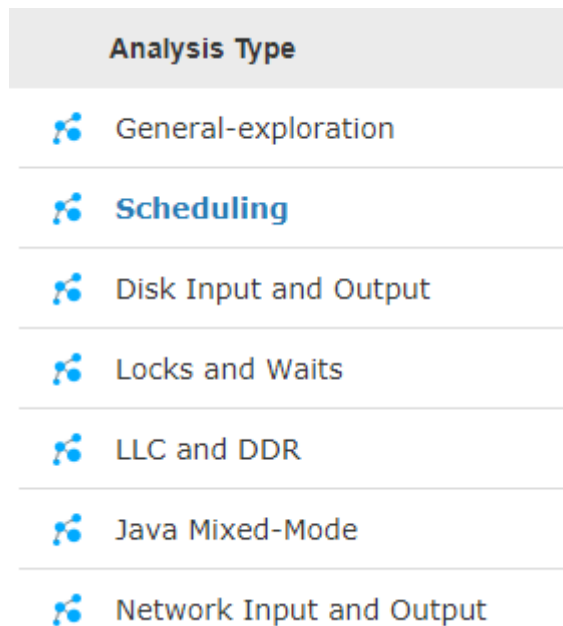


图 5.1.2

5.1.3 Disk IO 分析

Disk IO 分析，即磁盘输入输出分析方法，可选中分析方法 Disk Input and Output，如图 5.1.3。

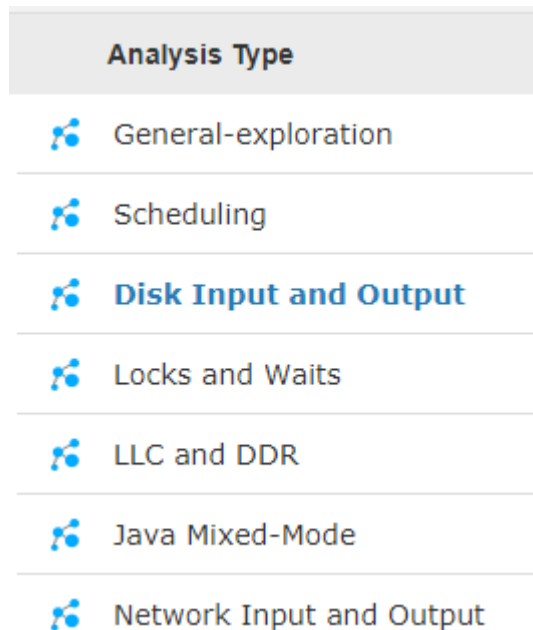


图 5.1.3

5.1.4 锁和等待分析

Locks and Waits 分析，即 glibc 锁和等待分析方法，包括 sleep、usleep、mutex、cond、spinlock、rwlock、semaphore 接口调用引起的并发性能分析，可选中分析方法 Locks and Waits，如图 5.1.4。








Analysis Type
 General-exploration
 Scheduling
 Disk Input and Output
 Locks and Waits
 LLC and DDR
 Java Mixed-Mode
 Network Input and Output

图 5.1.4

5.1.5 LLC&DDR 分析

LLC&DDR 分析，即分析 L3 Cache 和 DDR 内存的缓存命中率及读写的带宽信息。















Analysis Type
 General-exploration
 Scheduling
 Disk Input and Output
 Locks and Waits
 LLC and DDR
 Java Mixed-Mode
 Network Input and Output

图 5.1.5








5.1.6 Java Mixed-Mode 分析

Java Mixed-Mode 分析，即支持分析 java 程序的代码，但称之为 Mixed-Mode，是因为不仅仅能采集到 java 方法，同时还能采集到 native 代码。

Analysis Type	
	General-exploration
	Scheduling
	Disk Input and Output
	Locks and Waits
	LLC and DDR
	Java Mixed-Mode
	Network Input and Output

5.1.7 Network Input and Output 分析

Network Input and Output 分析，分析每个网口上的收发包统计信息，包括收发包带宽、收发包个数、收发包错包个数和收发包丢包个数。

Analysis Type	
	General-exploration
	Scheduling
	Disk Input and Output
	Locks and Waits
	LLC and DDR
	Java Mixed-Mode
	Network Input and Output

5.2 分析对象选择及参数配置

主要有三种形式的分析对象，分别为 Launch Application、Attach to Process、Profile system。

5.2.1 Launch Application

Launch Application 即采集启动的时候同时启动 **Application**（如图 5.2.1），采集时长受 **Application** 的执行时间来控制，适用于 **Application** 运行时间较短的场景。

参数配置：

Application 参数，配置被采集 **Application** 的绝对路径。

Application Parameters 参数，配置 **Application** 本身的执行参数。提供 **Modify** 功能。

Working directory 参数，配置 **Application** 的运行绝对路径，默认与 **Application** 的绝对路径一致。

CPU Sampling interval 参数，配置数据采样时间间隔，默认值 1。阈值范围[1,1000]。

Analysis system-wide 参数，勾选之后实现全系统采集。

CPU mask 参数，配置需要关注的 CPU 核。

Code profiling mode，选中代码分析模式。

Profiling LLC and DDR，勾选之后采集 LLC 和 DDR 驱动。

Modify 功能，即弹出较大的文本编辑框。

Launch Application

Specify and configure your analysis target: an application or a script to execute.

Application: *

Application Parameters: Modify

☒ Use application directory as working directory

Working directory: *

CPU Sampling interval (ms): *

☐ Analyze system-wide

CPU mask:

Code profiling mode:

☐ Profiling LLC and DDR

Command Line

```
malluma-analyzer --collect general-exploration --profiling-mode=native --interval=1 --
app-working-dir=/home/sampleApp/demo/matrix/linux --cpu-mask=0-30 --
/home/sampleApp/demo/matrix/linux/matrix.gcc
```

Binary/Symbol Search Modify

C/C++ Source Search Modify

图 5.2.1

5.2.2 Attach to Process

Attach to Process 即采集启动的时候 **Server** 中应用正在运行（如图 5.2.2），采集时长需要配置参数控制，适用于某些应用需要长时间持续运行的场景。

参数配置：

PID 参数，配置被采集应用的 **PID** 号。

Collection duration 参数，配置采集时长。

CPU Sampling interval 参数，配置数据采样时间间隔，默认值 1。阈值范围[1,1000]。

Profiling LLC and DDR，勾选之后采集 LLC 和 DDR 驱动。

Attach to Process

Specify the process to analyze. Performance data will be collected after attaching to the process.

PID/TID

32366*

Collection duration (s)

30*

CPU Sampling interval (ms)

1*

☐ Profiling LLC and DDR

Code profiling mode

Native

图 5.2.2

5.2.3 Profile System

Profile System 即采集整个 **Server OS** 系统（如图 5.2.3），无需关注系统中有哪些类型的应用在运行，采集时长需要配置参数控制，适用于多业务混合运行和有 **Child Process** 的场景。

参数配置：

Collection duration 参数，配置采集时长。

CPU Sampling interval 参数，配置数据采样时间间隔，默认值 1。阈值范围[1,1000]。

CPU mask 参数，配置需要关注的 CPU 核。

Profiling LLC and DDR，勾选之后采集 LLC 和 DDR 驱动。

Profile System

Configure setting for system-wide analysis. Select this analysis type to analyze system performance as a whole instead of paticular applications or processes.

Collection duration (s)

30*

CPU Sampling interval (ms)

1*

CPU mask

0-30

☐ Profiling LLC and DDR

图 5.2.3

5.2.4 命令行脚本展示

根据三种采集方式 **Launch Application**、**Attach to Process**、**Profile System** 的配置参数自动关联生成命令行采集脚本（如图 5.2.4），可以直接复制到命令行执行，支持新建采集和采集结果查看两个功能场景。

Profile System

Configure setting for system-wide analysis. Select this analysis type to analyze system performance as a whole instead of particular applications or processes.

Collection duration (s)

30

CPU Sampling interval (ms)

1

CPU mask

0-30

☐ Profiling LLC and DDR

Command Line

```
malluma-analyzer --collect general-exploration --interval=1 --cpu-mask=0-30 --system-wide --duration=30
```

图 5.2.4

5.3 分析启动及状态反馈

5.3.1 启动分析

完成分析对象和参数配置之后，点击 **Start** 按钮启动采集（如图 5.3.1.1），执行采集过程进入到 **Collection Log** 界面。



图 5.3.1.1

启动采集之前或采集之后，可以点击 **Binary/Symbol Search** 的 **Modify** 按钮（如图 5.3.1.2），弹出对话框配置带符号表的 **Application** 的路径或符号表的路径。配置符号表信息，适用于 **Application** 运行时不含符号表信息的场景。配置方式如下：

1) 用户配置的相对路径下面可以放带 **debug** 信息的二进制文件。例如，采集时运行的二进制应用文件所在路径为 **/home/prog1**，用户在弹出的对话框配置相对路径为 **/symbol**，则用户需要将带 **debug** 信息的 **prog1** 二进制文件放在服务器的 **/symbol/home/prog1** 路径。

2) 用户配置的相对路径下面可以放不带 **debug** 信息的二进制文件和与之对应的 **debuginfo** 文件。例如，采集时运行的二进制文件所在路径为 **/home/prog1**，用户配置相对路径为 **/symbol**，则用户需要将不带 **debug** 信息的 **prog1** 二进制文件放在服务器的 **/symbol/home/prog1** 路径，将 **debuginfo** 文件放在服务器的 **/symbol/home/debuginfo** 路径。

3) 用户配置符号表相对路径后，代码映射时，所有函数的代码映射都会在配置的路径下进行查找，在采集全系统的情况下，会涉及采集众多的二进制程序，每个二进制程序的函数如需要代码映射，都要在相对路径下进行 1) 或 3) 步描述的操作。

启动采集之前或采集之后，可以点击 **C/C++ Source Search** 的 **Modify** 按钮（如图 5.3.1.2），弹出对话框配置 APP 的源码路径。配置源码路径信息，以便查看源代码与 PMU 指标的映射关系。配置方式如下：

用户在配置的相对路径下面放置 **C** 源代码文件。例如，采集的应用程序编译时源文件所在路径为 **/home/prog1.c**，用户在弹出的对话框配置相对路径为 **/Source**，则用户需要将 **prog1.c** 文件放在服务器的 **/Source/home/prog1.c** 路径。

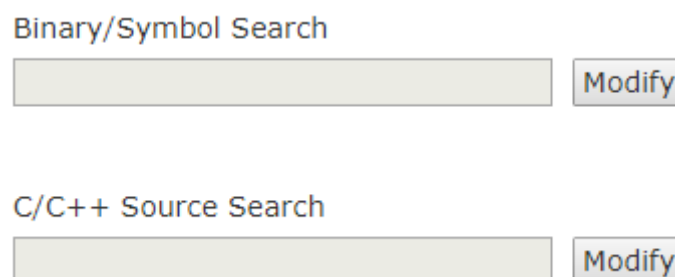


图 5.3.1.2

如果新建的分析 **Java Mixed-Mode** 类型或者在 **General Exploration** 分析类型里面选择 **code profiling mode** 为 **Java Mixed-Mode**，则会显示 **Java Source Search** 对话框（如图 5.3.1.3），这个时候要把需要映射源码的 **.java** 文件放到该目录下才能支持代码映射功能：



图 5.3.1.3

5.3.2 状态反馈

基于 **Collection Log** 功能，直接打印采集日志信息到界面（如图 5.3.2），反馈采集成功的状态和解析的日志信息，以及采集失败或解析失败的报错信息。支持新建采集分析和采集结果查看两种功能场景。

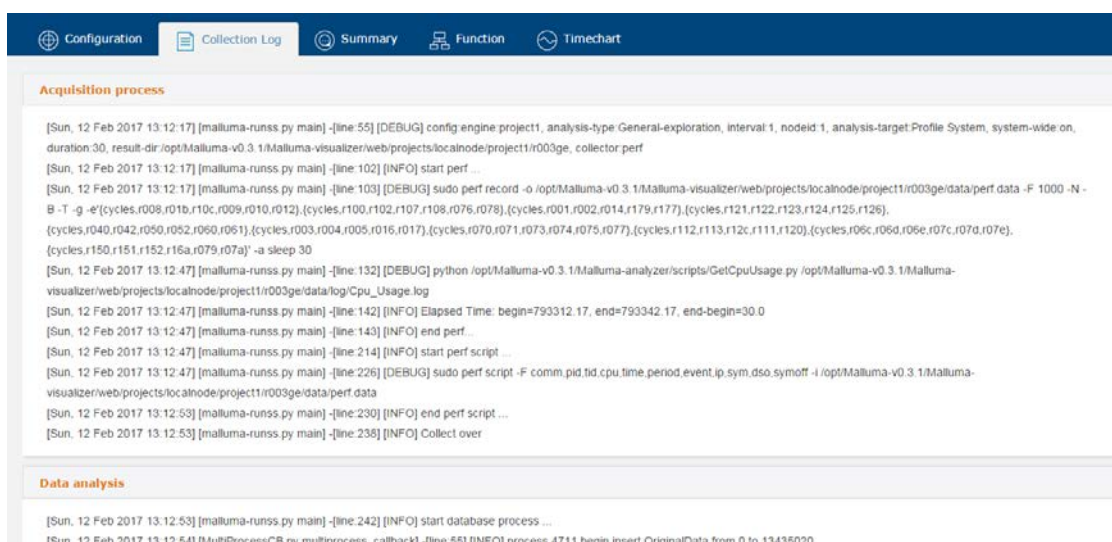


图 5.3.2

5.4 分析结果查看

5.4.1 Summary 分析

采集解析完成之后，生成 **Summary** 分析报告。

- **Top-down 整体分析**（如图 5.4.1.1）
提供 **Top-down** 模型的整体值。

Top-Down Total Results	
Elapsed Time(s)	40.320
Cycles	1164674694662
Instructions	7867338837
CPU_Active_Ratio	0.215
IPC	0.068
Retiring	0.023
+ Bad_Speculation	0.001
+ Frontend_Bound	0.003
+ Backend_Bound	0.974

图 5.4.1.1 对应 General-exploration 分析

- **Statistics 整体分析**（如图 5.4.1.2）

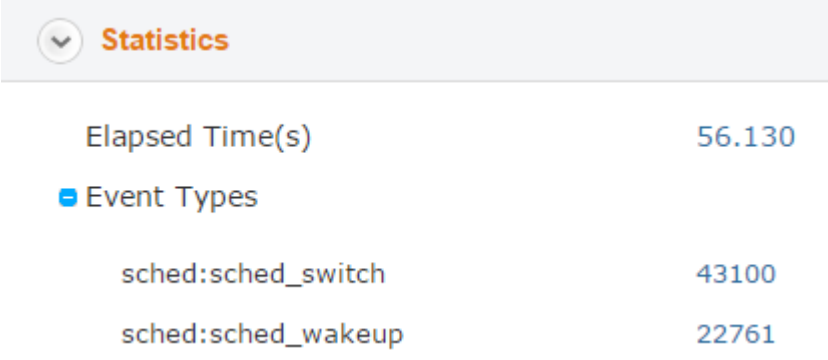


图 5.4.1.2 对应 Scheduling 分析

➤ Disk IO 整体时延统计分析（如图 5.4.1.3）

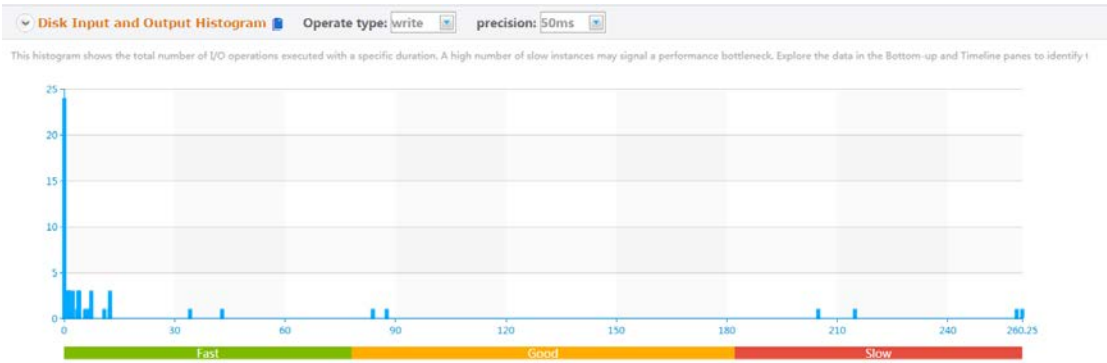


图 5.4.1.3 对应 Disk IO 分析

➤ Locks and Waits 统计分析（如图 5.4.1.4）（注：Wait Time 时间是所有线程等待时间的总和，在多线程情况下可能会大于程序执行时间）

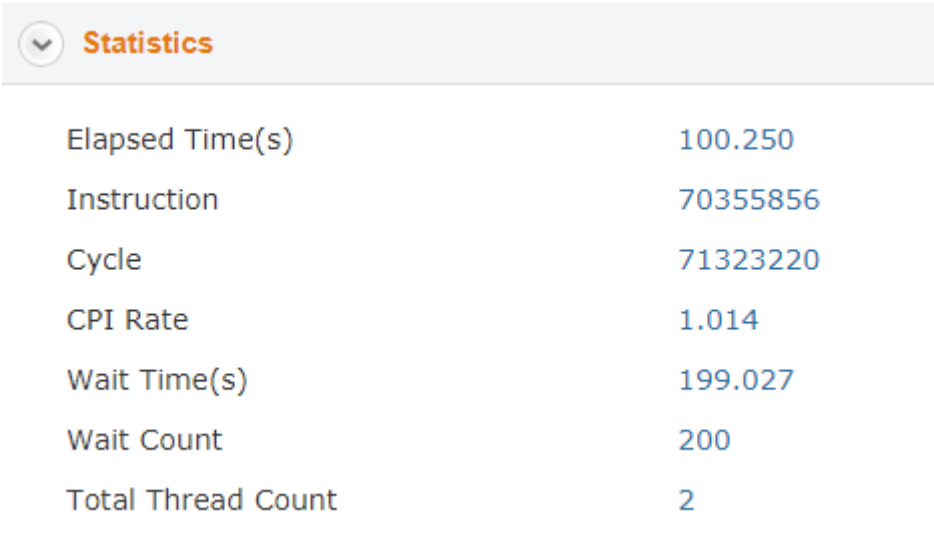


图 5.4.1.4 对应 Disk IO 分析

- 采集 OS 系统基本指标
 - CPU Usage（如图 5.4.2）
提供采集过程中 CPU Usage 的时序数据，以及平均值。

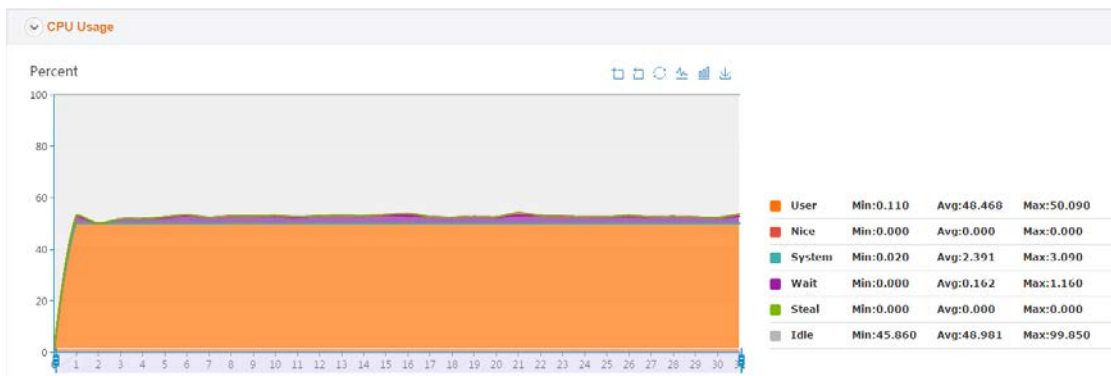


图 5.4.2

- 采集平台基本信息（如图 5.4.3）
提供采集执行 OS 系统的 CPU、OS、kernel 等环境信息，以及采集结果时间、结果大小等基本信息。

Collection and Platform Info	
Operating System:	4.1.44 Linux
Computer Name:	EulerOS
Result Size:	182.317MB
Collection start time:	2018-04-18 17:58:01
Collection end time:	2018-04-18 17:58:53
+ CPU1	
+ CPU2	

图 5.4.3

- 辅助功能介绍
导出 top-down 整体值到 xls 文件。

5.4.2 Function 分析

- 查看 Function 的 top-down 指标值（如图 5.4.4-5.4.8）
分析不同函数对应 top-down 模型的各指标值，按选定指标排序查看其 TopN 热点函数，可知针对某些 PMU 指标哪些函数是主要瓶颈。（注意：cycles 太小的函数，对应其他指标值可行度低。）

图 5.4.4图 5.4.5图 5.4.6

	Core/Function/Callstack	Cycles	Instructions	IPC	Retiring	Miss_Pred_Rate	Miss_Pred_BTBluBTB_Rate	PC_Write
1	total	72115142789	4002625911	0.555	0.185	0.016	0.011	0.106
2	lib 15	1793818651 (2.4736%)	1719879472 (42.969%)	0.964	0.321	0.008	0.009	0.176
3	lib 17	8748321552 (12.131%)	202299396 (5.054%)	0.231	0.077	0.026	0.010	0.050
4	lib 11	7546977103 (10.465%)	333467544 (8.331%)	0.442	0.147	0.021	0.000	0.103
5	lib 12	4409747786 (6.115%)	18690769 (4.669%)	0.424	0.141	0.023	0.010	0.088
6	lib 13	4037169678 (5.598%)	216205947 (5.402%)	0.536	0.179	0.061	0.027	0.043
7	lib 18	3478165689 (4.82%)	80205831 (2.004%)	0.231	0.077	0.021	0.012	0.048
8	lib 14	3240666802 (4.494%)	102017169 (2.549%)	0.315	0.105	0.023	0.016	0.084
9	lib 4	3167051711 (4.392%)	134440371 (3.359%)	0.424	0.141	0.003	0.012	0.093
10	lib 33	2336908133 (3.241%)	114369786 (2.857%)	0.489	0.163	0.037	0.019	0.063
11	lib 2	2136485072 (2.963%)	79252667 (1.96%)	0.371	0.124	0.021	0.008	0.103
12	lib 16	1800081451 (2.519%)	27170910 (6.79%)	0.170	0.057	0.031	0.034	0.039
13	lib 37	1440987527 (1.998%)	101640015 (2.539%)	0.705	0.235	0.009	0.004	0.106
14	lib 36	1335249030 (1.852%)	90089696 (2.251%)	0.675	0.225	0.042	0.014	0.186
15	lib 39	1296569381 (1.801%)	158713685 (3.955%)	1.222	0.407	0.011	0.049	0.115
16	lib 19	1042600159 (1.446%)	57692892 (1.441%)	0.553	0.184	0.026	0.025	0.104
17	lib 10	918268724 (1.273%)	47018447 (1.175%)	0.512	0.171	0.032	0.004	0.103
18	lib 32	782489593 (1.08%)	32927838 (8.23%)	0.421	0.140	0.039	0.026	0.111
19	lib 34	683348278 (0.948%)	40651350 (1.016%)	0.595	0.198	0.009	0.012	0.093
20	lib 35	647279763 (0.898%)	46612144 (1.165%)	0.720	0.240	0.013	0.020	0.175
21	lib 22	542277473 (0.752%)	19607167 (4.87%)	0.380	0.120	0.030	0.030	0.155
22	lib 38	513877358 (0.712%)	75056608 (1.875%)	1.461	0.487	0.015	0.027	0.120
23	lib 26	494867557 (0.672%)	4180137 (1.04%)	0.098	0.029	0.011	0.008	0.038

图 5.4.7

	Class/Method/Callstack	Cycles	Instructions	IPC	Retiring	Miss_Pred_Rate	Miss_Pred_BTBluBTB_Rate	PC_Write
1	total	72115142789	4002625911	0.555	0.185	0.016	0.011	0.106
2	lib unknown	6538016205 (9.063%)	1040866345 (26.005%)	1.219	0.406	0.013	0.007	0.221
3	lib LaunchUnsafe	1390387883 (1.928%)	7589023 (1.89%)	0.055	0.018			0.012
4	lib LaunchUnsafeTestWorkerThread	271916739 (0.377%)	15134638 (3.779%)	0.557	0.186			0
5	lib LaunchUnsafeTestWorkerThreadSupport	2419616 (0.003%)	0 (0%)	0	0			0
6	lib LaunchUnsafeSystem	0 (0%)	0 (0%)					

图 5.4.8

提供 4 种不同组合方式来查看，Core/Thread/Module/Function/Callstack 与 top-down 指标的关系（如图 5.4.8）。

- Grouping Function/Callstack
- Grouping Module/Function/Callstack
- Grouping Thread/Function/Callstack
- Grouping Core/Function/Callstack
- 对于采集 java 程序，还支持按照 Grouping class/Method/Callstack

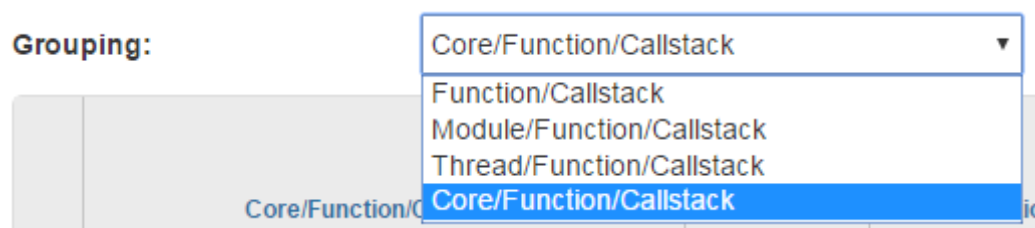


图 5.4.8

➤ 辅助功能

- Grouping 结果可以上下折叠
- 基于选中 PMU 指标排序
- PMU 指标可左右折叠
- 导出当前界面结果到 CSV 文件

5.4.3 Timechart 分析

- 查看 PMU 指标时序图（如图 5.4.9）
基于选中的 PMU 指标（最多 4 个），显示其时序图。可分析不同指标在整个数据采集过程中的波形走势，同一时刻不同指标的关联影响，以及选定一段时间内的 top-down 整体值、TopN 热点函数列表、波形等数据变化。

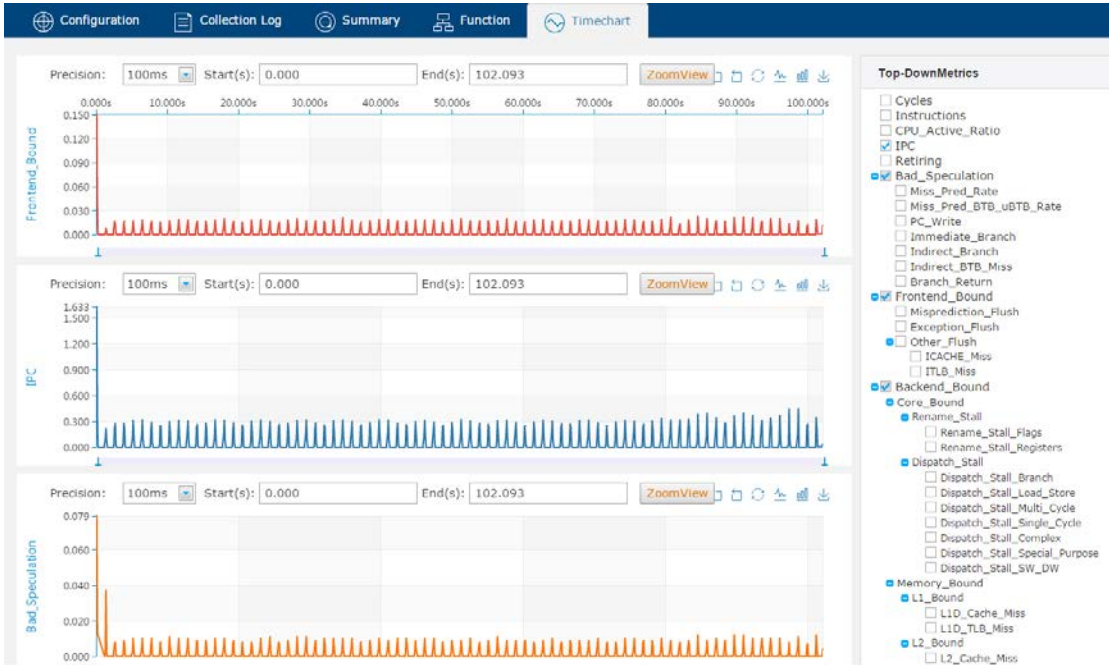


图 5.4.9

- 辅助功能
 - 基于任一 PMU 指标时序图窗口放大（如图 5.4.10），选中标记处鼠标左键在时序图中选中指定时间区域，自动放大此时间区域的时序图。

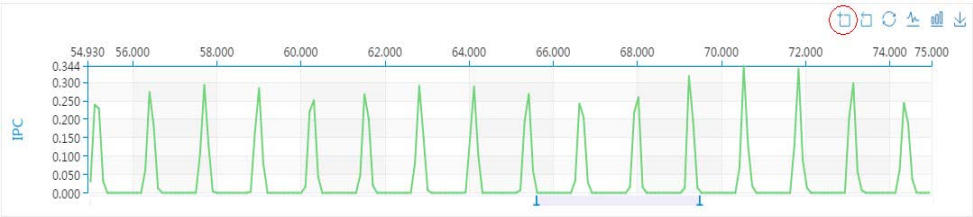


图 5.4.10

- 基于任一 PMU 指标时序图选择窗口滑动分析（如图 5.4.11），选中标记处鼠标左右拖动。

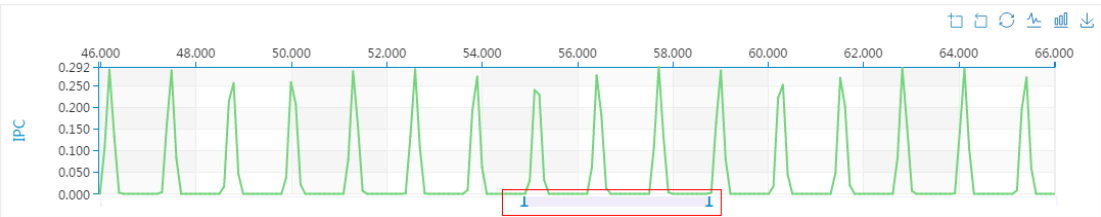


图 5.4.11

- 基于任一 PMU 指标时序图选择窗口过滤分析
- 波形图可以支持不同的插值步长，默认为 100ms（如图 5.4.12）。

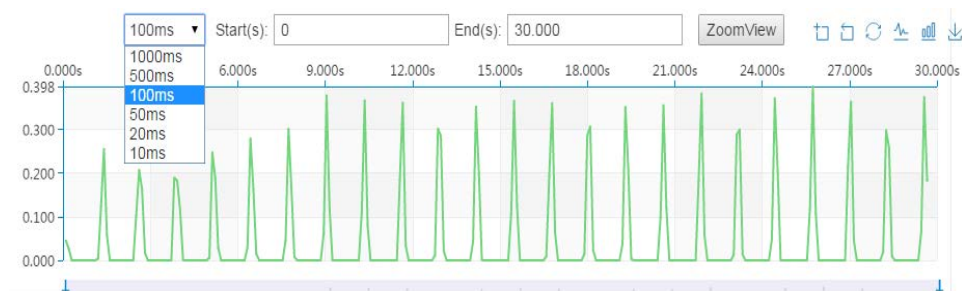


图 5.4.12

5.4.4 Zoomview 分析

- 查看圈定时间内的热点函数及指标时序图（如图 5.4.13）
在 Timechart 页签点击如下红圈中的 **zoom** 按钮，然后在波形图上鼠标左键框选时间段范围，再点击 **ZoomView** 按钮，会弹出 **Zoomview** 页签，并生成对应时间段的分析。

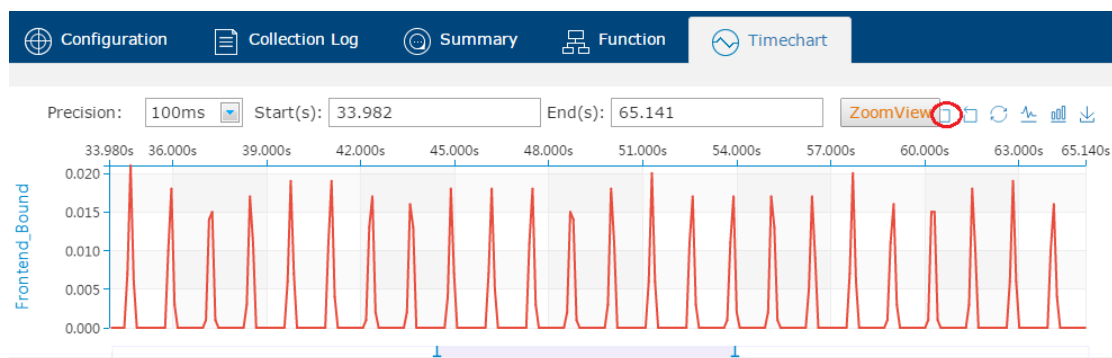


图 5.4.13

- **Zoomview 页签**
 - **Zoomview** 页签包括左侧的 **Zoom List** 区域，上部的 **Function** 区域和下部的 **Timechart** 区域（如图 5.4.14），其中 **Zoom List** 区域为选择的时间段对应的分析名称，如名称“time-33982-65141”表示从 15.822 秒到 19.172 秒这段时间内的分析，**Function** 区域为选择时间段内的热点函数，**Timechart** 区域为选择时间段的波形图。

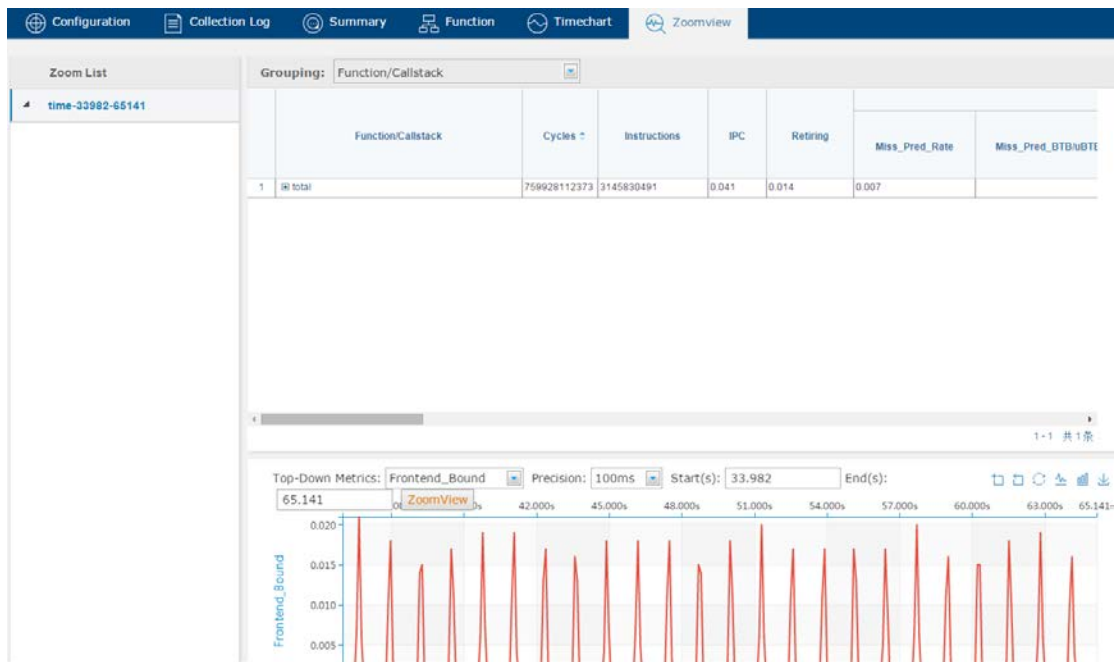


图 5.4.14

- 可以在 Zoomview 页签生成的波形图上，继续选择时间段生成次一级分析，方法跟 Timechart 页签生成时间段分析相同，此时的次一级分析名称显示在一级分析名称下面（如图 5.4.15）。

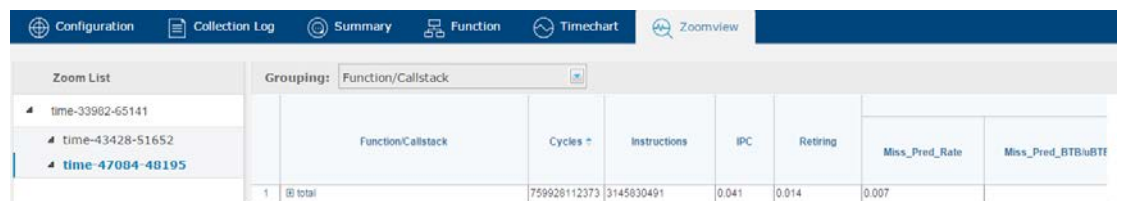


图 5.4.15

5.4.5 Code 分析

- 查看热点函数的代码映射（如图 5.4.16），通过该功能可以分析热点函数内部的热点指令，热点指令即指函数内 Cycles 事件占比最高的 Top 指令，功能还支持查看热点指令对应的高级语言文件及行号。功能还支持对汇编代码进行控制流分析，通过划分 basic block 并标示出跳转关系及颜色，可以清晰看到各个汇编代码块的“热度”。

在 Function 页签双击函数名称，会跳转到 Code 页签，Code 页签分为几个区域：

- 左侧为 Function List，是模块名及模块内的函数的列表
- 右侧上部包括：HardWare Event 下拉菜单，支持显示不同的 HardWare 事件，Total Count 表示该事件的求和值，File Name 表示函数所在的源文件
- 右侧下部包括函数源代码、函数汇编代码和函数汇编代码 basic block 的控制流图。在源代码和汇编代码区域，计算出每行源代码和汇编代码对应的 HardWare 事件计数值及占该事件 Total Count 的百分比（值为空的表示 0），百分比最高的即为函数的热点指令，在控制流图区域，通过图形标示出 basic block 的跳转关系和颜色，可以直观看到函数内的热点及调用关系。

在 Code 页签的源代码区域，包括：

- Source Line: 源代码行号
- Source: 改行对应的源代码
- Count(Percent): 该行源代码对应的 PMU 事件计数值及占该事件 Total Count 的百分比
- 过滤框: 过滤出包含指定字符串的源码

在 Code 页签的汇编代码区域，包括：

- Address: 汇编指令地址
- Source Line: 汇编指令对应的高级代码行号
- Assembly: 汇编指令
- Count(Percent): 该行汇编代码对应的 PMU 事件计数值及占该事件 Total Count 的百分比
- 过滤框: 过滤出包含指定字符串的汇编代码
- 折叠: 通过点击统一打开或者折叠所有 basic block

辅助功能

在源代码、汇编代码或控制流图上任意区域单击，三者可以联动高亮：

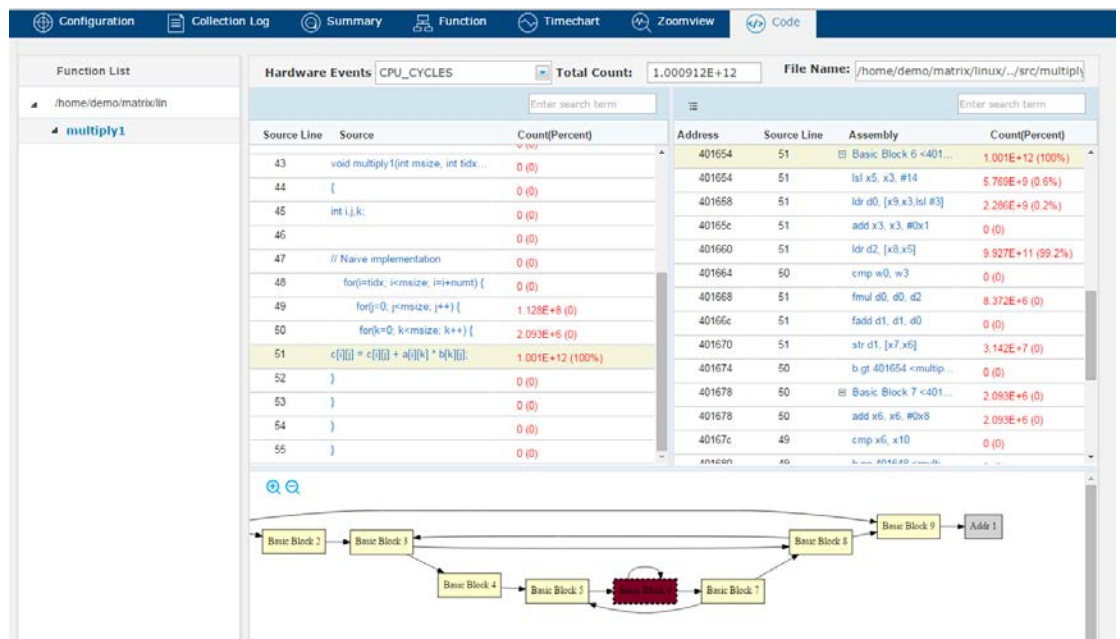


图 5.4.16

对于采样类型为 Java Mixed-Mode 采集到的 java 的方法，因为 java 的 jit 编译的限制，当前只有对应的源码，没有汇编和控制流图区域，如图 5.4.16.1：

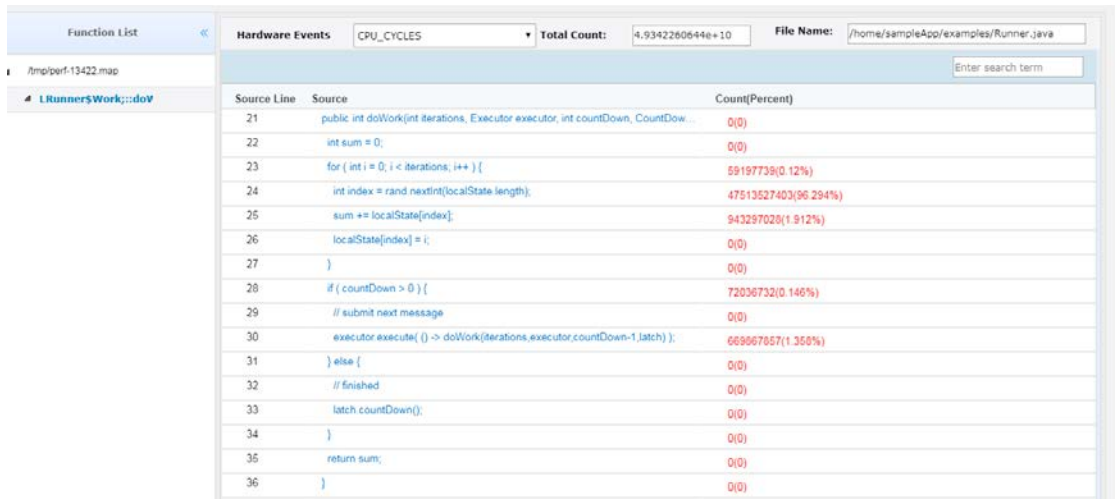


图 5.4.16.1

对于采集 java 程序，为了使能映射的源码更精确，建议开启-
XX:+UnlockDiagnosticVMOptions -XX:+DebugNonSafepoints 选项。

5.4.6 Resource 分析

- 查看采样时间后台 CPU 各个核的调度情况（如图 5.4.17）
显示采样时间内各个 CPU 核的运行调度情况，时间精度最小是微妙。将鼠标悬停在图上任意一点，显示当前时刻 CPU 核的运行状态，包括 idle 和 running 这 2 种状态，并显示状态起止时间和持续时间，对于 running 状态显示当前运行的进程名字和进程 id。其中 idle 状态表示 CPU 空闲，running 状态表示 CPU 正在运行程序。
- Filter 按钮：点击弹出过滤对话框，可以过滤掉不关注的 CPU 核，如图 5.4.18
- Function 按钮：选中 Function 按钮后，页签下面出来 Function 表格，显示选中的 CPU 在选中时间段内的热点函数
- reset 按钮：如图 5.4.17 红圈，时序图放大缩小后，点该按钮恢复原始精度
- 左/右箭头：沿着选中时间点及选中 CPU 核，向上或下一个状态单步前进
- 上/下箭头：固定选中时间点，在上或下一个 CPU 核上切换
- 放大/缩小按钮：放大/缩小时序图的，当前最小精度为微妙级

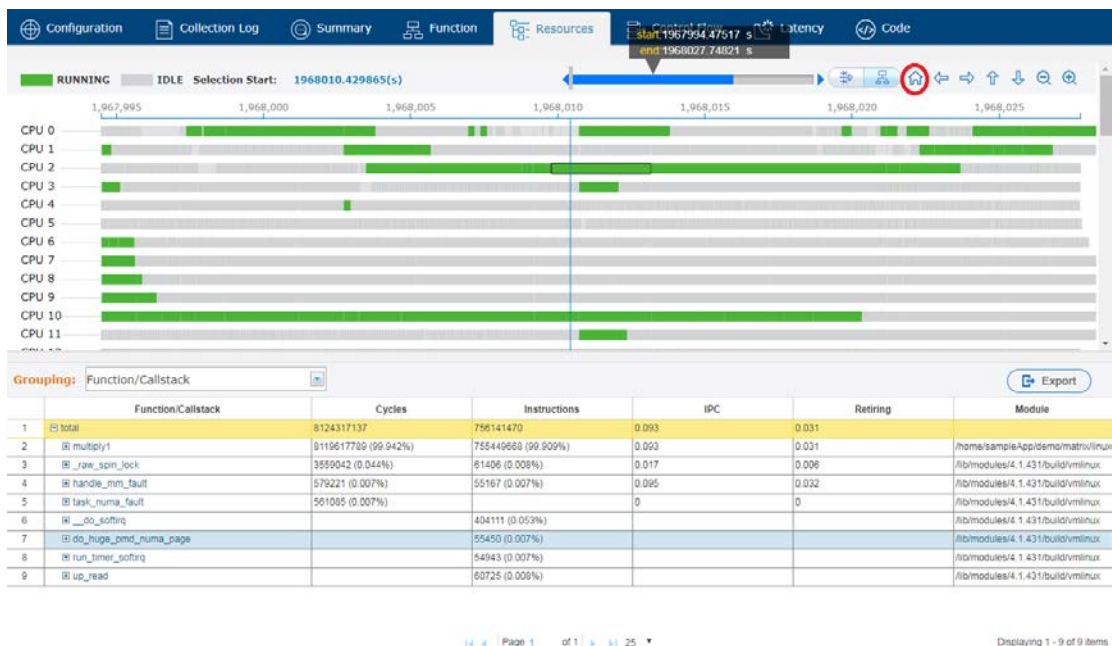


图 5.4.17

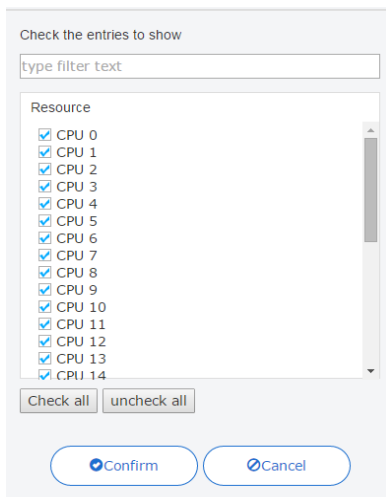


图 5.4.18

5.4.7 Control Flow 分析

- 查看采样时间后台进程的调度情况（如图 5.4.19）
显示采样时间内各个进程的调度情况，时间精度最小是微妙。将鼠标悬停在图上任意一点，显示当前时刻该进程的调度状态，包括 `wait_blocked`、`wait_for_cpu` 和 `running` 这 3 种状态，并显示状态起止时间和持续时间。其中 `wait_blocked` 状态表示进程阻塞无法运行，`wait_for_cpu` 状态表示进程处于可运行队列中，`running` 状态表示进程正在运行。
- **Filter** 按钮：点击弹出过滤对话框，可以过滤掉不关注的进程，如图 5.4.20
- **Function** 按钮：选中 **Function** 按钮后，页签下面出来 **Function** 表格，显示选中的线程在选中时间段内的热点函数
- **reset** 按钮：如图 5.4.19 红圈，时序图放大缩小后，点该按钮恢复原始精度

- 左/右箭头：沿着选中时间点及选中进程，向上或下一个状态单步前进
- 上/下箭头：固定选中时间点，在上或下一个进程切换
- 放大/缩小按钮：放大/缩小时序图的，当前最小精度为微妙级

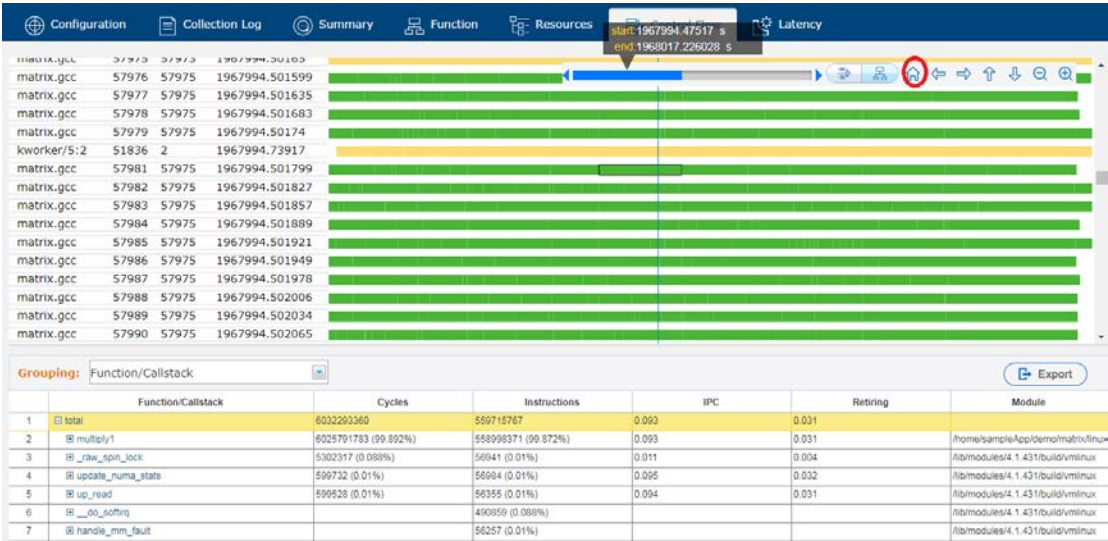


图 5.4.19

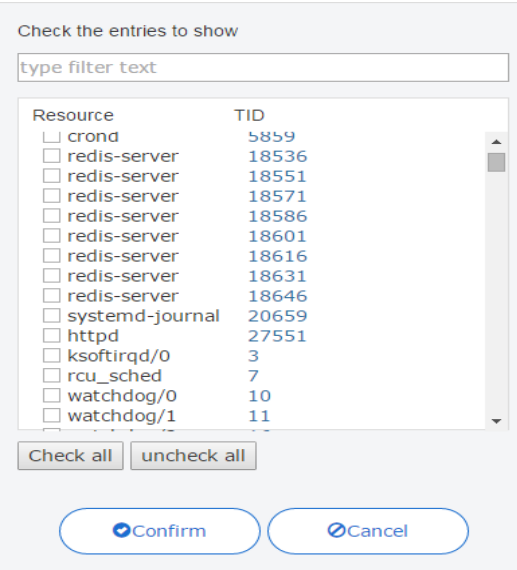


图 5.4.20

5.4.8 Latency 分析

- 查看采样时间后台进程的调度延迟情况（如图 5.4.21）
调度延迟即进程进入运行队列到获取处理器执行之间的时间差。可以统计采集期间所有进程的切换次数、平均调度延迟、最大调度延迟和最大延迟的时间点。
支持按照 Task 名字进行过滤和导出 excel 格式的数据。

⚙️ Configuration

📁 Collection Log

📄 Summary

🔧 Resources

📡 Control Flow

🕒 Latency

Task:

📄 Export

NO.	Task	Switches	Average delay (ms)	Maximum delay (ms)	Maximum delay at (s)
1	kworker/2:1:16589	34	3.996265	4.005999	4686080.888857
2	kworker/17:2:17052	5	3.995400	4.003000	4686080.708854
3	kworker/9:0:18355	3	3.992000	4.000000	4686080.808852
4	kworker/30:1:32040	22	3.991162	4.002000	4686076.552853
5	kworker/31:0:27620	8	3.986125	4.003000	4686078.540855
6	ksoftirqd/23:123	1	3.980000	3.980000	4686112.632866
7	ksoftirqd/21:113	3	3.976000	3.981000	4686108.224860
8	ksoftirqd/16:88	1	3.974000	3.974000	4686082.596854
9	ksoftirqd/30:158	2	3.963000	3.977000	4686110.516852
10	kworker/16:2:2556	62	3.935823	4.005000	4686121.264858

图 5.4.21

5.4.9 Disk IO 分析

- 查看采样时间 Disk IO 的情况（如图 5.4.22）

Thread: 查看采样时间每个线程的 I/O Wait 和 I/O APIs 时序图

I/O Queue Depth: 查看采样时间每个磁盘的 I/O 请求队列长度变化时序图

I/O Operation: 查看采样时间每个磁盘的 I/O 操作时序图

I/O Transfer: 查看采样时间每个磁盘的 I/O 操作数据大小时序图

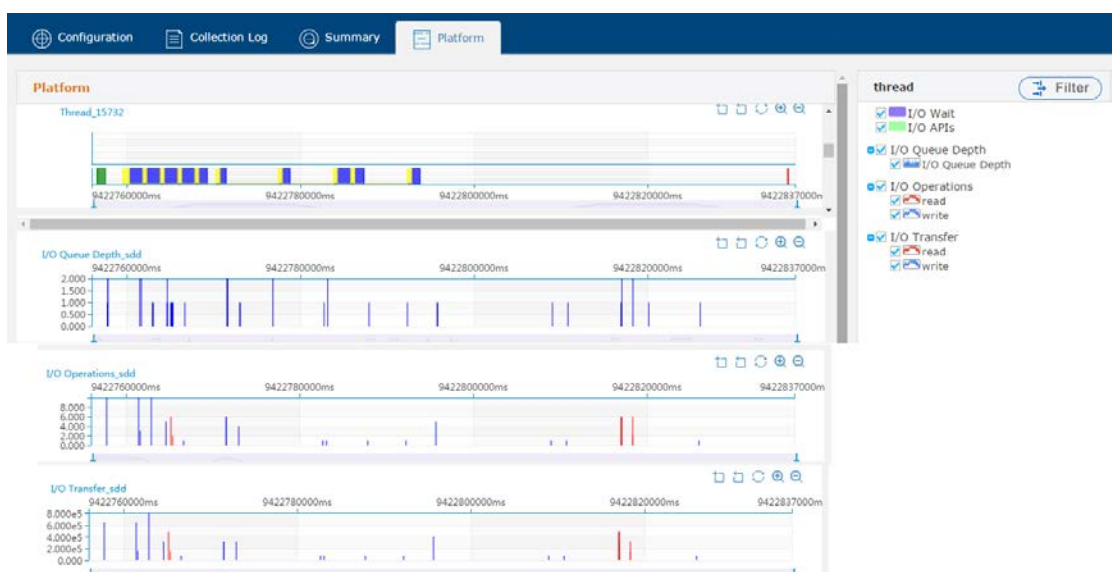


图 5.4.22

5.4.10 Locks and Waits 分析

- 查看采样时间程序锁和等待的情况（如图 5.4.23）

界面分为 2 部分，上部为表格，下部为时序图。上部的表格对采集到的数据从 Function/Callstack、SyncObject/Function/Callstack、Thread/Function/Callstack 和 Module/Function/Callstack 这 4 个维度进行分组展示，表格的 Wait Time 列表示等待时长，Wait Count 列表示等待次数。通过双击函数名，可以跳转到 Locks and Waits Code 页面。下部为进程/线程时序图，其中黄色表示线程因为调用 glibc 的 sleep、usleep、mutex、cond、

spinlock、rwlock 或 semaphore 接口导致进入了阻塞状态，通过鼠标悬浮于时序图上，可以看到引起程序阻塞的函数及其调用栈。

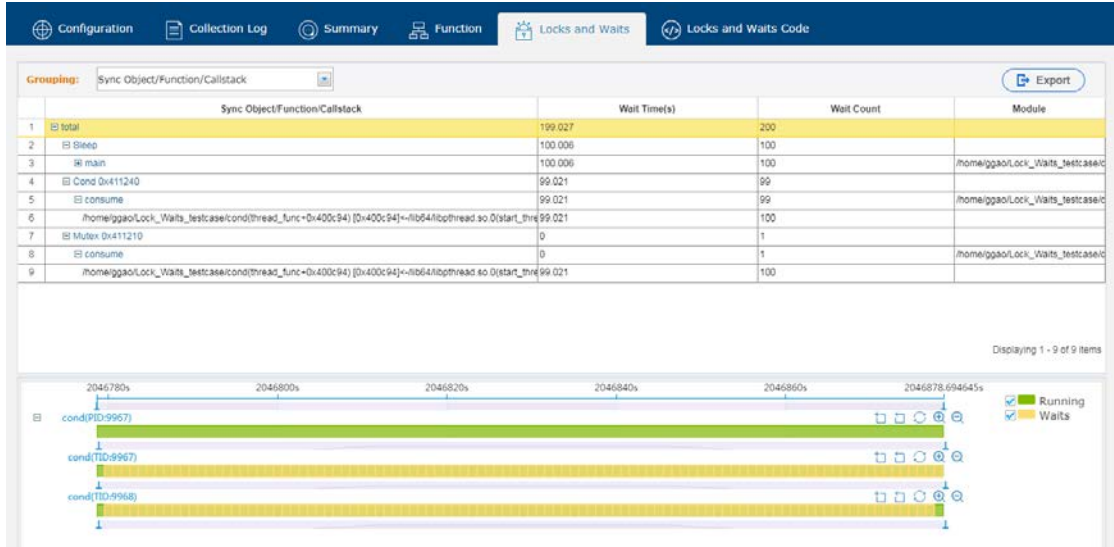


图 5.4.23

5.4.11 Locks and Waits Code 分析

➤ 查看采样时间程序锁和等待的情况（如图 5.4.24）

通过在 Locks and Waits 页面双击函数，可以跳转到该页面。该页面可以展示程序中调用了 glibc 的 sleep、usleep、mutex、cond、spinlock、rwlock 或 semaphore 接口的函数及其反汇编。如下，左侧源代码区域展示了该函数调用 sleep(1)阻塞的时间和次数，右侧汇编代码区域展示了跳转指令到 sleep 函数的阻塞时间和次数。

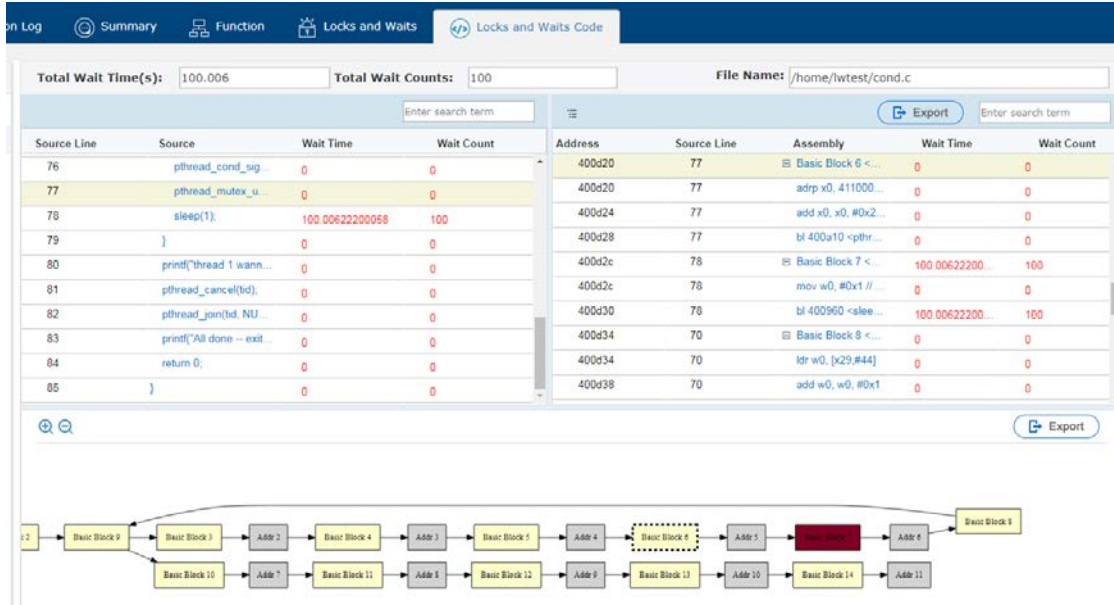


图 5.4.24

5.4.12 LLC&DDR 分析

注：为了使用 LLC&DDR 分析功能，需要首先编译、安装对应内核版本的内核驱动程序

➤ 查看采样时间 LLC&DDR 的汇总情况（如图 5.4.25）

下图展示了汇总的分析数据，指标包括 CPU 每个 die 的 llc 读写带宽及命中率，及 CPU 的 ddr 内存采集时间的读写带宽。

LLC and DDR statistics											
Item	Die0_Hit(MB/s)	Die0_Total(MB/s)	Die0_Hit_Rate	Die1_Hit(MB/s)	Die1_Total(MB/s)	Die1_Hit_Rate	Die2_Hit(MB/s)	Die2_Total(MB/s)	Die2_Hit_Rate	Die3_Hit(MB/s)	Die3_Total(MB/s)
l3c_rd	26.000	50.062	51.904	2200.093	3581.552	61.420	1039.544	1639.870	63.392	25.359	25.359
l3c_wr	2.967	4.044	73.364	8.757	14.213	61.612	6.307	8.184	77.064	2.206	2.206
ddr_rd		4.315			3431.848			12.126			
ddr_wr		3.087			6.132			7.715			

图 5.4.25

➤ 查看采样时间 LLC&DDR 的时间分布情况（如图 5.4.26）

下图展示了 CPU 每个 die 的 llc&ddr 的读、写命中率及带宽，左侧主轴表示带宽，单位是 MB/s，右侧副轴表示命中率，单位是百分比。

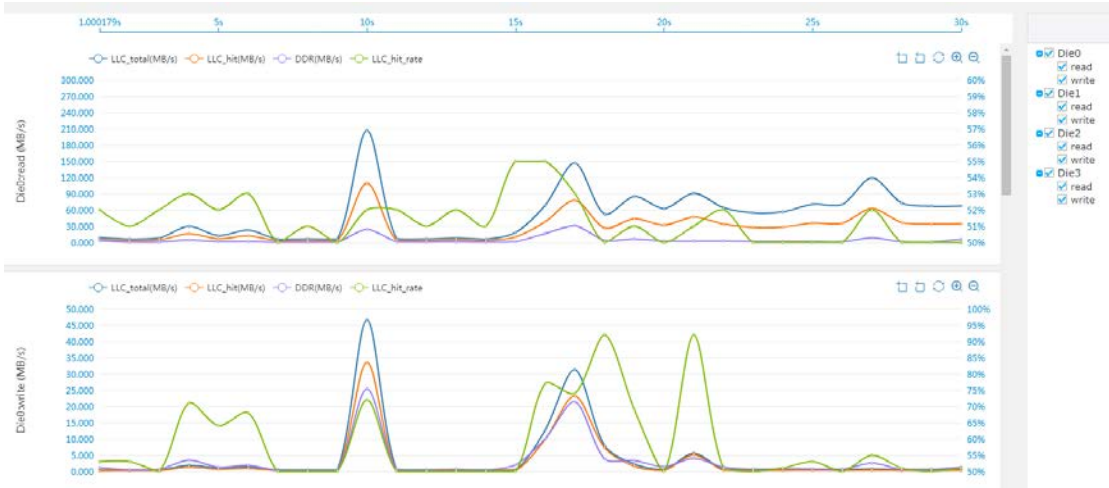


图 5.4.26

5.4.13 Flame Graph 分析

➤ 查看采样时间热点函数的火焰图（如图 5.4.27），帮助用户直观且迅速找到热点代码路径：

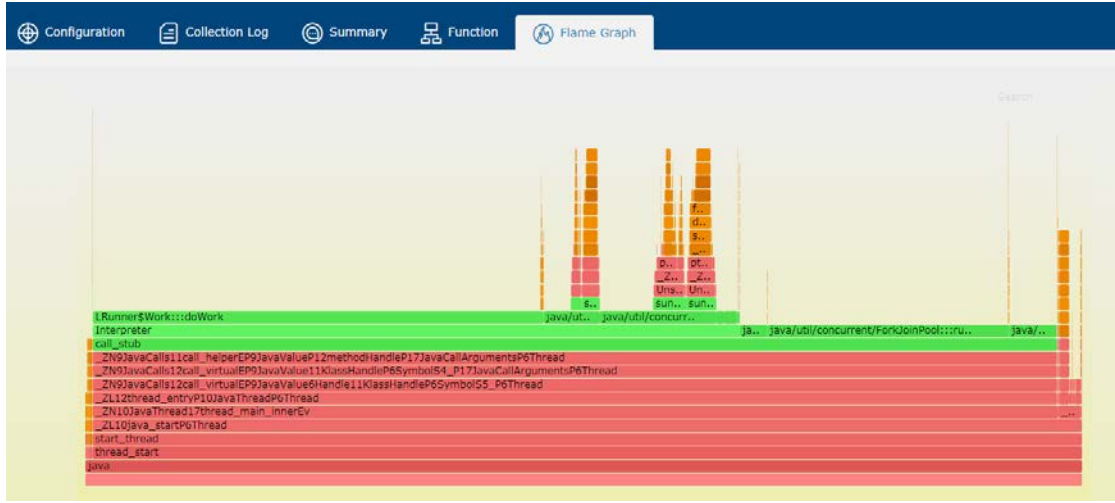


图 5.4.27

5.4.14 Network Input and Output 分析

- 查看采样时间 Network Input and Output 的汇总情况（如图 5.4.28）
下图展示了汇总的分析数据，指标包括每网口收发包带宽、收发包个数、收发包错包个数和收发包丢包个数。

Interfaces

item	eth2	eth3	lo
receive(MB/s)	188.211	0.104	0
receive_packets	2986418	913	0
receive_drop_packets	0	0	0
receive_error_packets	0	0	0
send(MB/s)	11168.381	0.017	0
send_packets	467463	151	0
send_drop_packets	0	0	0
send_error_packets	0	0	0

图 5.4.28

- 查看采样时间的 Network Input and Output 时间分布情况（如图 5.4.29）
下图展示了每个网口收发包带宽、收发包个数、收发包错包个数和收发包丢包个数随时间的变化情况，左侧主轴表示带宽，单位是 MB/s，右侧副轴表报文个数，单位是 packet。

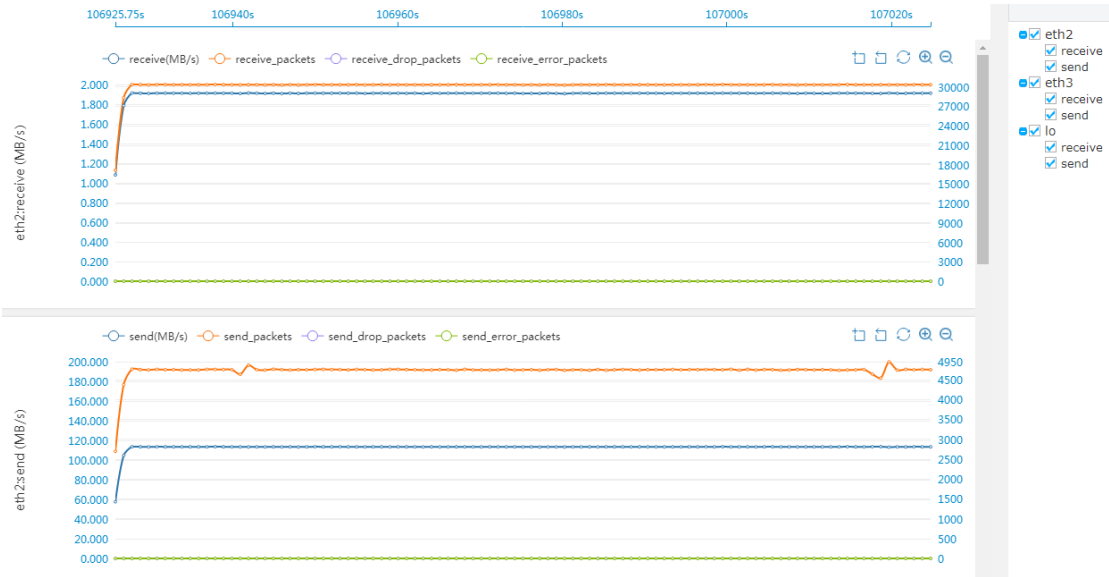


图 5.4.29

6 命令行界面支持

6.1 功能介绍

命令行工具 `malluma-analyzer` 可以直接在命令行进行参数配置执行数据采集和简单分析（如图 6.1），进一步可以导入采集结果到图形界面进行详细分析。适用于无法直接连接 web 的场景。

```
server for tools/develop: /home/ggao/Malluma-v0.60718/Malluma-analyzer/bin64 # python malluma-analyzer --help
Usage: malluma-analyzer [options] [--] <application> [<args>]

Example:
1.Collection general-exploration data and report summary.
  malluma-analyzer <--collect general-exploration> [-r ./r@ge] <a.out> [param1 param2 ]
  malluma-analyzer <--collect general-exploration> [-r ./r@ge] <-p $pid> <-d 30>
  malluma-analyzer <--collect general-exploration> [-r ./r@ge] <--system-wide> <-d 30>
  malluma-analyzer <--collect general-exploration> [-r ./r@ge] <--system-wide> [--cpu-mask 2,5-8] <-d 30>
2.Collection scheduling data and report latency.
  malluma-analyzer <--collect scheduling> [-r ./r@sch] <a.out> [param1 param2 ]
  malluma-analyzer <--collect scheduling> [-r ./r@sch] <-d 30>
3.Collection diskio data.
  malluma-analyzer <--collect diskio> [-r ./r@diskio] <--system-wide> <-d 30>
  malluma-analyzer <--collect diskio> [-r ./r@diskio] <-p $pid> <-d 30>
  malluma-analyzer <--collect diskio> [-r ./r@diskio] <--interval=1> <--app-working-dir=/app-dir> <-- app> [param1 param2...]
4.Collection locks and waits data.
  malluma-analyzer <--collect locksandwaits> [-r ./r@locksandwaits] <-p $pid> <-d 30>
  malluma-analyzer <--collect locksandwaits> [-r ./r@locksandwaits] <--interval=1> <--app-working-dir=/app-dir> <-- app> [param1 param2...]
5.Report summary and print top-down total results to the screen.
  malluma-analyzer <--report summary> <-r ./r@ge>
6.Report scheduling and print latency results to the screen.
  malluma-analyzer <--report scheduling> <-r ./r@sch>

Options:
-h, --help            show this help message and exit

Collection options:
--collect=ANALYSIS_TYPE
                        Specify analysis type, example. --collect general-
                        exploration or --collect scheduling.
--analysis-target=ANALYSIS_TARGET
-r DIRECTORY, --result-dir=DIRECTORY
                        Specify the directory used for creating the data
                        collection results,example ./r000ge or ./r000sch.
-p PID, --target-pid=PID
                        Specify ID for the process to which data collection
                        should be attached.
--system-wide          Specify system for data collection.
--cpu-mask=STRING      Specify which CPU(s) <string> to collect data on. For
                        example, specify "2-8,10,12-14" to sample only CPUs 2
                        through 8, 10, and 12 through 14.
-d SECONDS, --duration=SECONDS
                        Specify duration for the collection in seconds.
-I MILLISECONDS, --interval=MILLISECONDS
                        Specify an interval of data collection (for example,
                        sampling) in milliseconds. Selectable range [1,1000].
--app-working-dir=host
                        Specify application working dir.
--target-install-dir=host
                        Specify target installation folder.
--report=REPORT        Specify report type, example. --report summary or
                        --report scheduling.
-v, --version          Print version information.
```

图 6.1

提供数据采集功能和输出简单 Summary 分析报告功能（如图 6.2），详细采集命令和脚本示例见图 6.2、图 6.3。

```

arm44:/home # python /opt/Malluma/Malluma-analyzer/bin64/Malluma-analyzer.py --collect general-exploration sleep 5
Top-down total results:
-----
Elapsed Time(s)                5.09
Cycles                        959467
Instructions                   81076
CPU_Active_Ratio               0.0
IPC                           0.845
Retiring                       0.282
Bad_Speculation                0.019
  Miss_Pred_Rate               0.048
  Miss_Pred_BTB_uBTB_Rate      None
  PC_Write                     0
  Immediate_Branch             0
  Indirect_Branch              0
  Indirect_BTBT_Miss           None
  Branch_Return                0
Frontend_Bound                 0.904
  Misprediction_Flush          0.512
  Exception_Flush              0.002
  Other_Flush                   0.39
  ICACHE_Miss                  None
  ITLB_Miss                    None
Backend_Bound                  -0.204
  Core_Bound                   0
    Rename_Stall               0
      Rename_Stall_Flags       0
      Rename_Stall_Registers   0
    Dispatch_Stall             0
      Dispatch_Stall_Branch     0
      Dispatch_Stall_Load_Store 0
      Dispatch_Stall_Multi_Cycle 0
      Dispatch_Stall_Single_Cycle 0
      Dispatch_Stall_Complex    0
      Dispatch_Stall_Special_Purpose 0
      Dispatch_Stall_SW_DW      0
  Memory_Bound                 0
    L1_Bound                   0
      L1D_Cache_Miss            None
      L1D_TLB_Miss              None
    L2_Bound                   0
      L2_Cache_Miss             None
      L2_TLB_Miss               None
    L1D_Bandwidth(MB/s)         0
    L2_Bandwidth(MB/s)          0
    Bus_Bandwidth_RD(MB/s)      0
    Bus_Bandwidth_WR(MB/s)      0
-----
arm44:/home #

```

图 6.2


```
server-for-toolsdevelo:/opt/Malluma-v0.4/Malluma-analyzer/bin64 # python malluma-analyzer --collect scheduling -d 5
```

Task	Switches	Average delay ms	Maximum delay ms	Maximum delay at
kworker/28:1H:3659	1	avg: 0.000010 ms	max: 0.000010 ms	max at: 4698748.569617 s
kworker/3:1:17753	1	avg: 0.000009 ms	max: 0.000009 ms	max at: 4698744.876867 s
kworker/17:2:17052	1	avg: 0.000009 ms	max: 0.000009 ms	max at: 4698744.708864 s
kworker/28:1:4461	5	avg: 0.000008 ms	max: 0.000017 ms	max at: 4698745.580868 s
kworker/20:2:27716	1	avg: 0.000008 ms	max: 0.000008 ms	max at: 4698744.672865 s
kworker/23:2:13329	1	avg: 0.000008 ms	max: 0.000008 ms	max at: 4698744.632865 s
perf:3884	2	avg: 0.000007 ms	max: 0.000009 ms	max at: 4698743.700034 s
kworker/5:0:9613	4	avg: 0.000007 ms	max: 0.000008 ms	max at: 4698747.848858 s
kworker/6:2:24831	1	avg: 0.000007 ms	max: 0.000007 ms	max at: 4698744.840858 s
kworker/4:0:11829	1	avg: 0.000007 ms	max: 0.000007 ms	max at: 4698744.864868 s
kworker/24:0:29220	1	avg: 0.000007 ms	max: 0.000007 ms	max at: 4698744.620867 s
kworker/2:1:16589	1	avg: 0.000007 ms	max: 0.000007 ms	max at: 4698744.888865 s
kworker/21:2:28590	1	avg: 0.000007 ms	max: 0.000007 ms	max at: 4698744.656867 s
kworker/u64:2:1779	6	avg: 0.000007 ms	max: 0.000008 ms	max at: 4698744.524863 s
kworker/16:2:2556	6	avg: 0.000007 ms	max: 0.000008 ms	max at: 4698743.700863 s
kworker/25:0:7889	2	avg: 0.000006 ms	max: 0.000007 ms	max at: 4698744.608863 s
kworker/19:2:14151	1	avg: 0.000006 ms	max: 0.000006 ms	max at: 4698744.684860 s
khugepaged:570	1	avg: 0.000006 ms	max: 0.000006 ms	max at: 4698746.720857 s
kworker/7:1:29364	2	avg: 0.000006 ms	max: 0.000008 ms	max at: 4698745.828864 s
kworker/1:1:5318	2	avg: 0.000005 ms	max: 0.000007 ms	max at: 4698744.900866 s
kworker/8:1:9803	5	avg: 0.000005 ms	max: 0.000006 ms	max at: 4698745.580857 s
kworker/0:1:17402	2	avg: 0.000005 ms	max: 0.000007 ms	max at: 4698744.912864 s
watchdog/15:81	2	avg: 0.000005 ms	max: 0.000005 ms	max at: 4698744.248852 s
watchdog/5:31	2	avg: 0.000005 ms	max: 0.000005 ms	max at: 4698744.208854 s
watchdog/2:16	2	avg: 0.000005 ms	max: 0.000006 ms	max at: 4698744.196855 s
kjournald:3664	4	avg: 0.000004 ms	max: 0.000006 ms	max at: 4698748.576917 s
watchdog/25:131	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.288854 s
irqbalance:5470	1	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698744.624029 s
migration/6:37	1	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698743.700025 s
kworker/18:1:27051	2	avg: 0.000004 ms	max: 0.000007 ms	max at: 4698745.696860 s
watchdog/16:86	2	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698744.252854 s
watchdog/4:26	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.204856 s
watchdog/21:111	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.272854 s
watchdog/24:126	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.284856 s
watchdog/0:10	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.188856 s
watchdog/20:106	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.268856 s
watchdog/3:21	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.200854 s
watchdog/27:141	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.296854 s
watchdog/29:151	2	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.304853 s
iostat:3883	5	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698744.550118 s
kworker/22:2:31206	5	avg: 0.000004 ms	max: 0.000005 ms	max at: 4698744.644856 s
watchdog/1:11	2	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698744.192854 s
watchdog/18:96	2	avg: 0.000004 ms	max: 0.000004 ms	max at: 4698744.260853 s

图 6.3

6.2 命令参数

➤ 数据采集 Collect 相关:

参数	说明
--collect=ANALYSIS_TYPE	Specify analysis type, example. --collect general-exploration or --collect scheduling.
-r DIRECTORY, --result-dir=DIRECTORY	Specify the directory used for creating the data collection results,example ./r000ge or ./r000sch.
-p PID, --target-pid=PID	Specify ID for the process to which data collection should be attached.
--system-wide	Specify system for data collection.
--cpu-mask=STRING	Specify which CPU(s) <string> to collect data on. For example, specify "2-8,10,12-14" to sample only CPUs 2 through 8, 10, and 12 through 14.
-d SECONDS, --duration=SECONDS	Specify duration for the collection in seconds.
-l MILLISECONDS, --interval=MILLISECONDS	Specify an interval of data collection (for example, sampling) in milliseconds.
--app-working-dir=\$dir	Specify application working dir.

<code>--target-install-dir=\$dir</code>	Specify target installation folder.
<code>-v, --version</code>	Print version information.

➤ 分析报告 Report 相关:

参数	说明
<code>--report=REPORT</code>	Specify report type, example. <code>--report summary</code> or <code>--report scheduling</code> .
<code>-r DIRECTORY, --result-dir=DIRECTORY</code>	Specify the directory used for creating the data collection results, example <code>./r000ge</code> or <code>./r000sch</code> .

6.3 示例

6.3.1 服务器数据采集

Usage: `malluma-analyzer [options] [--] <application> [<args>]`

Example:

1.Collection data and report summary.

```

malluma-analyzer <--collect general-exploration> [-r ./r@ @ @ge] <a.out> [parm1 parm2 ]
malluma-analyzer <--collect general-exploration> [-r ./r@ @ @ge] <-p $pid> <-d 30>
malluma-analyzer <--collect general-exploration> [-r ./r@ @ @ge] <--system-wide> <-d 30>
malluma-analyzer <--collect general-exploration> [-r ./r@ @ @ge] <--system-wide> [--cpu-mask 2,5-8] <-d 30>

```

2.Report summary and print top-down total results to the screen.

```

malluma-analyzer <--report summary> <-r ./r@ @ @ge>

```

注: <>必选参数, []可选参数

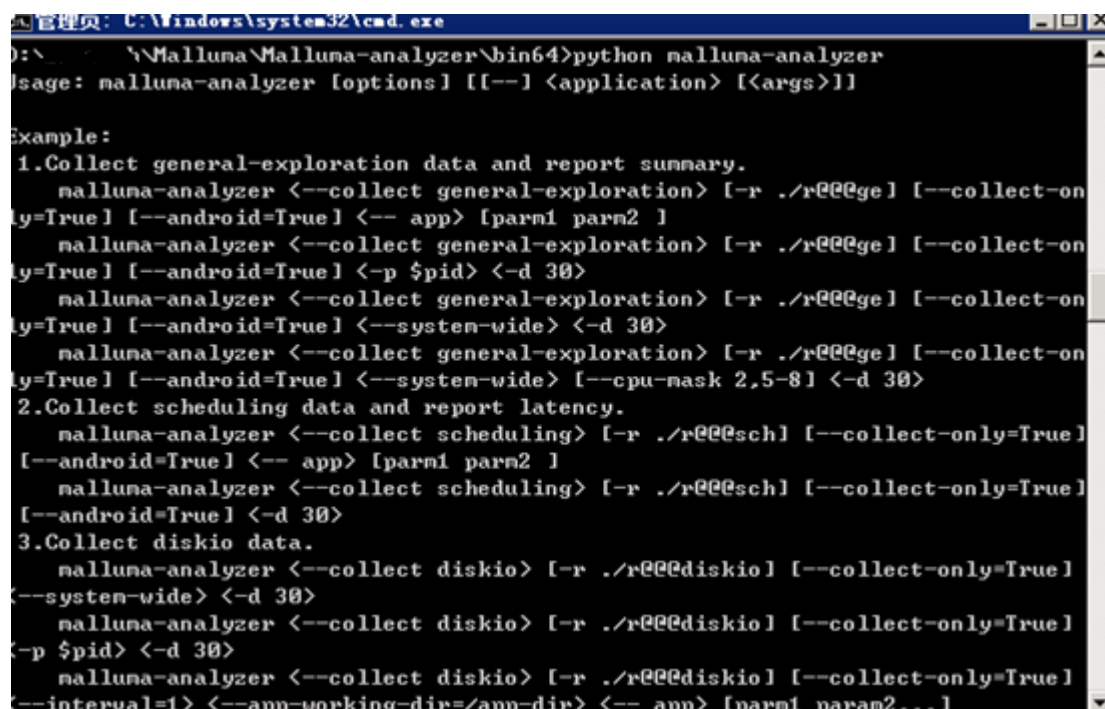
核心采集脚本样例:				
<code>malluma-analyzer</code>	<code>--collect</code>	<code>general-exploration</code>	<code>--</code>	#采集应用 matrix.gcc
<code>/home/xxxxxx/matrix/linux/matrix.gcc</code>				
<code>malluma-analyzer</code>	<code>--collect</code>	<code>general-exploration</code>	<code>-- taskset -c 2-20</code>	#采集应用 matrix.gcc,并绑定 2-20 核
<code>/home/xxxxxx/matrix/linux/matrix.gcc</code>				
<code>malluma-analyzer</code>	<code>--collect</code>	<code>general-exploration</code>	<code>-r /home/test/r000ge --</code>	#采集应用 matrix.gcc,并指定采集结果路径
<code>/home/xxxxxx/matrix/linux/matrix.gcc</code>				
<code>malluma-analyzer</code>	<code>--collect</code>	<code>general-exploration</code>	<code>-r /home/test/r000ge --app-working-dir /home/test -- /home/xxxxxx/matrix/linux/matrix.gcc</code>	#采集应用 matrix.gcc,并指定应用执行路径
<code>malluma-analyzer</code>	<code>--collect</code>	<code>general-exploration</code>	<code>-p \$pid -d 5</code>	#采集 PID, 5s
<code>malluma-analyzer</code>	<code>--collect</code>	<code>general-exploration</code>	<code>-p \$pid sleep 5</code>	#采集 PID, 5s
<code>malluma-analyzer</code>	<code>--collect</code>	<code>general-exploration</code>	<code>--system-wide -d 5</code>	#采集全系统, 5s
<code>malluma-analyzer</code>	<code>--collect</code>	<code>general-exploration</code>	<code>--system-wide sleep 5</code>	#采集全系统, 5s
<code>malluma-analyzer</code>	<code>--collect</code>	<code>general-exploration</code>	<code>--system-wide</code>	#采集全系统, matrix.gcc 执行结束采集完成
<code>/home/xxxxxx/matrix/linux/matrix.gcc</code>				

```

malluma-analyzer --collect general-exploration --system-wide --cpu-mask 2,5-8 -d 5 #采集指定核, 5s
malluma-analyzer --collect general-exploration --system-wide --cpu-mask 2,5-8 #采集指定核, 5s
sleep 5
malluma-analyzer --report summary -r r000ge #输出 summary 报告

```

6.3.2 Android 手机终端数据采集



```

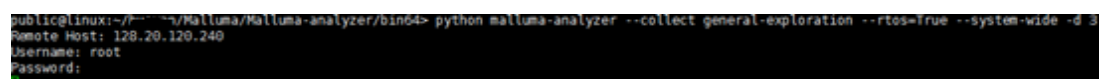
C:\Windows\system32\cmd.exe
D:\Malluma\Malluma-analyzer\bin64>python malluma-analyzer
Usage: malluma-analyzer [options] [--] <application> [<args>]

Example:
1. Collect general-exploration data and report summary.
   malluma-analyzer <--collect general-exploration> [-r ./r000ge] [--collect-only=True] [--android=True] <-- app> [parm1 parm2 ]
   malluma-analyzer <--collect general-exploration> [-r ./r000ge] [--collect-only=True] [--android=True] <-p $pid> <-d 30>
   malluma-analyzer <--collect general-exploration> [-r ./r000ge] [--collect-only=True] [--android=True] <--system-wide> <-d 30>
   malluma-analyzer <--collect general-exploration> [-r ./r000ge] [--collect-only=True] [--android=True] <--system-wide> [--cpu-mask 2,5-8] <-d 30>
2. Collect scheduling data and report latency.
   malluma-analyzer <--collect scheduling> [-r ./r000sch] [--collect-only=True] [--android=True] <-- app> [parm1 parm2 ]
   malluma-analyzer <--collect scheduling> [-r ./r000sch] [--collect-only=True] [--android=True] <-d 30>
3. Collect diskio data.
   malluma-analyzer <--collect diskio> [-r ./r000diskio] [--collect-only=True] <--system-wide> <-d 30>
   malluma-analyzer <--collect diskio> [-r ./r000diskio] [--collect-only=True] <-p $pid> <-d 30>
   malluma-analyzer <--collect diskio> [-r ./r000diskio] [--collect-only=True] <--interval=1> <--app-working-dir=/app-dir> <-- app> [parm1 param2...]

```

pc 通过 usb 连接手机后, 在 pc 上执行 malluma-analyzer 脚本, 传入参数--target-platform=android, 则脚本会通过 adb 自动连接手机执行采集, 采集的数据需要通过服务器导入在 web 界面进行分析查看。

6.3.3 rtos 单板数据采集



```

public@linuxr:~/Malluma/Malluma-analyzer/bin64> python malluma-analyzer --collect general-exploration --rtos=True --system-wide -d 3
Remote Host: 128.20.120.240
Username: root
Password:

```

在可以 ssh 到单板的服务器上, 执行 malluma-analyzer 脚本, 传入参数--target-platform=rtos, 则脚本会提示输入单板的 ip、用户名和密码, 脚本自动建立 ssh 链接后, 会向单板发送采集指令, 单板采集完成后数据需要通过服务器导入在 web 界面进行分析查看。