





专注云原生 引领新 IT

云监控平台(HC-CloudMonitor)产品说明书



目录

谐云科技云监控解决方案	错误!未定义书签。
目录	2
第1章 公司简介	6
1.1. 公司介绍	6
1.2. 团队介绍	7
1.3. 融资情况	8
1.4. 资产情况	8
第2章 云监控产品概述	9
2.1. 产品背景和价值	9
2.2. 云内应用访问流量采集器介绍	10
2.3. 功能架构	11
2.4. 技术架构	15
2.5. 部署架构图	17
2.6. 案例展示	17
第3章 云监控产品功能介绍	19
3.1. 总览	20
3.1.1. 业务概览	20
3.1.2. 新增应用列表	21
3.1.3. 集群概览	22



3.1.4.	告警概览2	3
3.2. 资源	〔监控2	4
3.2.1.	机房资源监控2	5
3.2.2.	集群资源监控2	8
3.2.3.	主机资源监控3	5
3.2.4.	命名空间监控3	9
3.2.5.	服务监控4	2
3.2.6.	工作负载监控5	1
3.2.7.	pod 监控5	3
3.2.8.	容器监控5	6
3.2.9.	管理组件监控	8
3.3. 业务	-监控5	9
3.3.1.	业务列表5	9
3.3.2.	业务节点列表6	3
3.3.3.	业务详情6	5
3.3.4.	节点详情6	9
3.3.5.	租户拓扑7	9
3.3.6.	归并8	0
3.4. 应用	1监控	4
3.4.1.	应用列表	4
3.4.2.	应用实例	6



3.4.3.	拓扑视图
3.4.4.	应用详情91
3.5. 告誉	
3.5.1.	告警事件114
3.5.2.	告警规则114
3.5.3.	通知对象117
3.5.4.	通知对象组118
3.6. 仪表	盘119
3.6.1.	面板列表119
3.6.2.	创建仪表盘120
3.6.3.	创建文件夹122
3.6.4.	导入122
3.6.5.	添加数据源123
3.7. 网络	
3.7.1.	DNS 异常124
3.8. 检索	
3.8.1.	日志收集126
3.8.2.	事件检索127
3.8.3.	日志检索128
3.8.4.	资源检索129
3.8.5.	操作审计130



3.9. 工具
3.9.1. tcp dump
3.10. 配置
3.10.1. 用户管理132
3.10.2. 租户管理
3.10.3. 全局配置
3.10.4. 集群配置
3.10.5. 业务配置
3.10.6. 应用配置140
3.10.7. 组件监控
3.10.8. 磁盘清理



第1章 公司简介

1.1. 公司介绍

杭州谐云科技有限公司成立于 2016 年 7 月,核心团队来自浙江大学超大规模信息系统实 验室,团队从 2001 年开始与美国道富银行合作,为道富银行重构和建设了新一代股票交易系 统等系统,赢得了华尔街的赞誉。2008 年开始历时两年为美国道富银行建设了私有云并承载 数百应用系统。陈纯院士看到了云计算在各行业广泛的应用前景,决定以超大规模系统研究中 心为依托成立浙江大学软件工程实验室 (SEL),专业从事开源云计算相关研究工作。实验室 由杨小虎、蔡亮教授领导,丁轶群博士、苌程博士等四名青年教师具体负责,并组织了四十余 位博士、硕士研究生共同研发云计算平台相关技术,并于 2013 年,成为 Cloud Foundry 中国 唯一两家核心代码贡献组织,在 2015 年成为成为谷歌发起的云原生计算组织 Cloud Native Computing Foundation (CNCF)创始成员。谐云与浙江大学建立了战略合作关系,确保公司人 才与技术始终保持创新。

谐云核心团队在云原生领域和应用支撑云平台系统上研究近十载,技术积累雄厚,并有多 个国内大型云平台落地实践,是国内云原生领域的布道者,并出版了国内第一本深度分析 Docker 技术的专业书籍《Docker:容器与容器云》。核心团队活跃在国际顶级开源社区,其中 为 Kubernetes 等项目贡献代码 1400 多万行,代码贡献量排行国内第一,国际第四。

谐云科技是国内为数不多掌握底层核心技术的容器云产品及解决方案提供商,专注于国产 自主可控的企业级智能化容器云平台(观云台),超强的技术研发实力获得知名云计算厂商、中 大型金融机构、电信运营商和能源,化工和制造业龙头企业的认可,目前产品在三十多家合作

第6页



伙伴落地。谐云容器云平台不仅面向传统数据中心提升研发运维效率,同时谐云也积极探索在物联网领域,边缘设备和边缘服务器等场景下轻量级容器平台的应用。

1.2. 团队介绍

公司核心团队来自于浙江大学 SEL 实验室,公司员工 400+。谐云团队在云计算及相关领 域,具备丰富的研发背景、深厚的技术积淀和国际化的视野。公司员工全部拥有大学及本科以 上学历,其中技术团队成员拥有硕士及以上学历者达到 70%以上,谐云作为浙江大学产学研战 略重点扶植云计算前沿企业,获得浙大人才方面的大力支持。

陈纯(董事长),教授,中国工程院信息与电子工程学部院士,兼任浙江省计算机学会理事长。

杨小虎(副董事长), 教授, 浙江大学计算机软件研究所副所长, 互联网金融研究院副院长。

蔡亮(董事),教授,浙江大学软件学院副院长,浙江省重大科技专项专家,分布式系统与 信息安全。

王翱宇(总经理),毕业于浙江大学,曾就职于道富银行、浙大网新,拥有12年以上金融信息化系统建设经验,在网新期间为公司创造了数亿元以上的营收,获CEO特殊贡献奖。

丁轶群(首席技术官),毕业于浙江大学,浙大 SEL 实验室创始人&带队老师。浙江省第一 批青年科学家。《Docker 容器与容器云》主要作者。设计建设国外高性能外汇交易系统、大型 电商云平台,智慧城市云平台,具备丰富高性能分布式系统工程经验。



苌程(副总裁),毕业于浙江大学,曾就职于道富银行、浙大网新,有7年以上大型金融系统研发经验,分布式追踪、分布式处理专家,主持研究上海交易所轻量级消息中间件和负责分布式事件处理引擎机制、国内 PaaS 平台首批实践者。

才振功(副总裁):毕业于浙江大学,主要从事云计算与智能化运维相关技术研发,先后承 担和参与了阿里巴巴前沿技术研发合作及与 IBM、CFETS 等国内外知名企业在智能运维领域的 联合研发项目,提出了面向企业级数据中心的智能故障治理、性能优化及容量规划解决方案, 申请和完成发明专利 5 项。

1.3. 融资情况

公司注册资本 2013.647058 万元,总部位于杭州,分支机构包含:深圳,郑州,北京,上海,成都等。

2016年9月,获城云科技,信雅达和风旗投资1500万人民币Pre-A轮融资。 2017年12月,获新湖智脑,如般量子和兰石投资数千万元人民币A轮融资。 2020年1月,获阿里巴巴数千万元人民币B轮融资。

1.4. 资产情况

序号	类型	名称
1	软件著作权证书	谐云应用性能监控 APM 软件 v1.0
2	软件著作权证书	谐云容器云平台系统软件 v1.0
3	软件著作权证书	谐云在线用户行为分析 UAM 软件 v1.0
4	软件著作权证书	谐云 SDN 交换机控制平台软件 v1.0
5	软件著作权证书	谐云应用系统性能容量规划平台软件 V1.0
6	软件著作权证书	谐云应用支撑系统软件 V1.0
7	软件著作权证书	谐云持续集成系统软件(简称:谐云 CI 系统) V1.0
8	软件著作权证书	谐云开旗服务平台软件 V1.0



9	软件著作权证书	谐云网络性能管理系统软件 V1.0
10	软件著作权证书	谐云微服务支撑平台软件 V1.0
11	软件著作权证书	谐云项目生命周期流程管理软件 V1.0
12	软件著作权证书	谐云业务流程监控系统软件 V1.0
13	软件著作权证书	谐云移动端性能监控系统软件 V1.0
14	软件著作权证书	谐云智能异常检测平台 V1.0
15	软件著作权证书	谐云自动化运维平台软件 V1.0
16	软件产品评估证书	谐云应用性能监控 APMv1.0
17	软件产品评估证书	谐云容器云平台系统软件 v1.0
18	软件产品评估证书	谐云在线用户行为分析 UAM 软件 v1.0
19	软件产品评估证书	谐云应用系统性能容量规划平台软件 V1.0
20	CMMI3	软件能力成熟度集成模型
21	ISO9001	管理体系认证证书
22	ISO27001	信息安全管理体系认证证书
23	ISO20000	信息技术服务管理体系认证证书
24	ISO14001	环境管理体系认证证书
25	ISO18001	职业健康安全管理体系认证证书
26	双软企业认证	
27	杭州市高新技术企业认证	
28	国家高新技术企业已认证	
29	CNCF 基金会	
30	可信云认证	
31	KCSP 认证	1.9/1.12/1.13/1.14 (k8s 版本)
32	КТР	CNCF 官方 KTP 培训合作伙伴

第2章 云监控产品概述

2.1. 产品背景和价值

目前市场上已经有一定规模的企业建设了容器私有云,私有云的弹性架构让应用更具灵活性、弹性和扩展性、在提高应用的效率的同时,也让应用的拓扑架构和通讯变得更复杂,更难监控。目前云体系市场上成熟方案普罗米修斯监控只具备最基础的资源监控功能,企业运维如



今缺乏手段获得容器内部的应用状态以及容器内部应用的东西流量访问情况。同时企业缺乏清晰的云上应用访问可见性,运维在云上也缺乏手段排查来自应用本身或者不同应用程序之间复杂交互的业务故障。

谐云云监控平台能够根据云上东西访问流量,动态识别云上复杂的应用交互关系,构建云 上业务拓扑运维图,动态掌握云上业务运行状态,解决现阶段缺乏业务视角对云上运行应用整 体宏观的监控以及缺乏业务应用之间故障定界手段;云监控平台同时也集成了 prometheus 和 grafana,关联业务指标和平台指标,实现平台问题和业务问题的定界;在深入问题定位方面, 云监控平台提供的动态 java 弹性探针,在不重启应用的前提下,加强捕获应用运行时性能问 题的能力,实现云上性能代码级监控。

2.2. 云内应用访问流量采集器介绍

谐云云监控的应用流量交互采集器是业界唯一能做到轻量级、高性能、低消耗全量获取云 内应用访问流量的采集器。它采用使用 ebpf 技术获取系统调用,通过分析系统调用获取应用 访问指标。hcmine 采集器和当前主流的网络流量镜像方式有着本质上的区别。当前主流的网 络探针使用的是 pcap 等网络包复制拦截技术,原理是将被监控机器上的指定端口的网络数据 包完全复制一份,然后模拟 linux 网络协议栈进行解包,其镜像流量和模拟协议栈解析数据包 需要的资源和应用本身消耗的资源几乎是等同的,就相当于把应用的资源消耗翻倍了,这对于 任何应用架构来说都是难以接受的。而使用分析 linux 系统调用记录的方式的网络探针 hcmine, 是从内核空间内获取到 linux 系统的系统调用记录,然后过滤出 linux 内核在处理网络数据时 的相关系统调用事件,再经过数据分析便可获取完整的云上应用访问流量,构建完整的东西流 量访问图。



2.3. 功能架构

谐云云监控平台主要利用云主机探针(hcmine)实时采集云上应用访问数据,构建云上应 用东西向访问关系;其次利用弹性应用探针动态对故障应用加入代码级别数据的采集,提供细 粒度问题定位手段;然后对采集的数据和普罗米修斯采集的指标数据进行统一分析,最终建立 以业务为出发点、融合容器监控,提供云上业务访问的可观测性,同时进一步增强运维云上业 务访故障追踪能力。



云监控功能架构图

主要功能项如下,弹性探针对应的应用分析以及平台功能参考下面章节功能描述。

模块名称	功能名称	功能点
<u>总览</u>	总览	支持该项目下所有业务集群告警等数据的统计展示。包括 业务概览,新增应用概览,集群概览,告警概览四大模块。 业务概览可展示该项目下所有业务的指标数据包括反映 业务健康程度的健康分数。新增应用概览可展示相应时间 段内新增的应用信息,支持跳转配置。集群概况分为容器 云及主机云两个维度,分别展示各自维度下集群的资源及 状态指标。告警概览为项目告警的数量及告警登记的统一 展示。
<u>资源</u>	k8s 集群	支持在页面展示包括 k8s 集群的资源及集群状态信息,包括 CPU,内存,磁盘,GPU 等资源信息



	主机集群	支持在页面展示包括传统主机集群的资源及集群状态信息,包括 CPU,内存,磁盘,GPU 等资源信息
	资源列表	支持在页面展示包括机房,机柜,服务器及集群组件,pod 在内的集群资源。
	3D 机房	支持在页面以 3D 的形式展示机房机柜服务器之间的 3D 建 模视图。
	全局应用拓扑	支持在页面以拓扑形式呈现当前集群内应用之间的交互 关系,同时支持多条件筛选应用拓扑
	业务列表	支持呈现不同时间区间的业务状态 支持通过业务名称、健康度对业务进行筛选 支持用户在全局应用拓扑上,新增、编辑业务
		支持按照租户视角呈现拓扑调用包括夸租户甚至跨集群 调用,展示各个租户节点的状态与告警 支持节点间呈现网络流量、延时数据,并按照延时呈现状
业务监控	租户拓扑	态 支持分析业务节点之间请求的具体调用,包括发送端,接 收端,发送数据量,发送耗时,接收数据量,接收耗时, 服务端处理时间等指标
		支持按照时间控件选择不同时间区间的聚合数据 支持回溯功能,实现对业务拓扑的历史状态追踪 支持按集应用服务两个维度横向展示 支持按集群命名空间归并等维度纵向分组聚合展示
	业务拓扑	支持按照业务视角呈现业务拓扑,展示各个业务节点的状态与告警 支持回溯功能,实现对业务拓扑的历史状态追踪
		文持百满功能, 实现为亚劳拍针的历史状态追踪 支持按集应用服务两个维度横向展示 支持按集群命名空间归并等维度纵向分组聚合展示 支持分析业务节点之间请求的具体调用,包括发送端,接
		收端,发送数据量,发送耗时,接收数据量,接收耗时, 服务端处理时间等指标
		文持按照时间注户选择不问时间区间的乘百数缩 支持下钻查看详细告警、业务状态指标趋势图 支持与系统分层信息功能联动,按照相同业务视角,呈现 系统分层信息拓扑
	系统信息分析(应	支持以分层拓扑展示应用依赖的运行时环境,用户能够可 视化查看 k8s 或者虚拟机运行环境的资源使用情况



	用环境依赖分析)	支持用户点击分层故障信息,下钻到对应节点的详情,呈 现节点详情信息、告警等
		支持按照不同时间维度,筛选业务 - 应用的状态,支持再 拓扑图上呈现消亡、存活的节点
		支持点击全屏,呈现拓扑
	仪表板概览	支持进入到该页面,默认呈现容器云集群 dashboard,用户 能够总览的查看整个容器云集群的概览
容器监控	仪表板列表	dashboard 的汇总列表,支持呈现默认所有的 dashboard,并 支持对 dashboard 进行管理
<u></u>	自定义仪表	支持创建监控 dashboard、修改监控 dashboard、删除监控 dashboard、查看监控 dashboard 内的图表
	盘	支持通过 Prometheus、mysql、ES、OpenTSDB 等方式导入 其他数据源,自定义仪表盘
	拓扑视图	支持在拓扑视图中默认显示当前项目最近 30 分钟内的应 用拓扑情况 支持全局拓扑及调用时间、节点异常显示 支持节点归并及下钻 支持列表展示调用关系及节点状态
	监控概览	支持展示 30mins 内实例中每个请求的响应时间散点图 支持平均执行时间、访问的总次数(吞吐量)、apdex 指 数和访问该应用的错误率的监控
	应用拓扑	支持应用调用链路完整展示,包括外部服务、负载均衡、 中间件,异步调用并展示节点状态,支持下钻 支持节点异常及调用时间的监控
	事务追踪	支持展示指定时间区间内耗时最长的 100 个 Web 事务图表 及其响应时间和吞吐量 支持查看 URL 总览中的某个事务详情,并在慢事务追踪栏 目显示其中最慢的几次事务 支持请求的 trace 信息显示 支持每条请求中慢事务显示及分析 支持历史慢方法显示 支持慢堆栈慢方法抓取 支持软硬件及网络关联分析 支持通过 transactionid 来查看异常分析,显示该条链路的 拓扑图和详细信息
	异常分析	展示实例中所有发生的异常及详情信息, error 级别及 debug 级别
	慢方法	展示应用中抓取的全部慢方法,含框架慢方法、业务慢方 法



		支持慢方法调用链功能 支持展示所选时间段最慢的 5 个慢方法的平均响应时间		
亡日下令		吞吐量		
<u> </u>	数据库详情	支持展示该应用所访问数据库的详情情况,数据库包括 mysql、redis、mongodb等 支持图表形式展示平均耗时最长的 100 条 SQL 语句,可根 据响应时间和吞吐量进行排序 支持单条 SQL 事务的响应时间和吞吐率 支持慢 SQL 追踪及下钻分析 展示实例中的 nosql 及慢 nosql 的 trace 信息		
	INOSQI	文 行 展 示 mango、 red is 和 cassandra 二 种 尖 空 的 nosql , 有 到 平 均 耗 时 和 吞 吐 量 , 可 切 换 为 图 表 展 示		
	服务调用	展示所有了调用类型(内部、外部),包括调用的错误率、 吞吐量和响应时间,有助于分析服务质量 支持展示平均响应时间最久和错误率最高的服务调用曲 线图 支持查看调用服务的详情、响应时间、吞吐率,支持下钻 分析		
	消息队列监控	支持展示应用中消息队列整体情况及慢消息的详情 支持从生产者和消费者的不同角度展示消息队列的地址、 消息总数、每分钟消息数和平均消息发送时间(消费者为 平均消息处理时间),整体把握消息队列的健康情况		
	JVM 概况	支持 jdk5 及以上的 jvm 内存池监控 支持线程信息、死锁线程展示、实例信息、GC 算法监控		
	线程剖析	支持按时间段下载线程的 jstack 信息 支持设定未来时间下载线程的 jstack 信息		
	用户管理	用户可在此处新增人员并分配权限,可修改,可删除		
	租户管理	用户可在此处新增租户并分配权限,可修改,可删除		
配置中心	探针管理	支持应用探针的下载 支持查看已部署探针的列表信息,支持对探针进行休眠及 包追踪开关操作		
	集成管理	支持对 Rancher 系统的对接,支持从 Rancher 同步集群信息		
	全局配置	支持对组件地址, 部署地, 数据中心相关配置的设置		
	集群配置	支持对进程探活相关进行配置,可修改		
	业务配置	支持对慢阈值,健康评分策略,健康检查,url 过滤功能进 行配置		



1		
	应用配置	支持设置项目下应用的网络探针开关、应用探针过滤异常 类型及数据处理参数 支持配置项目下不同应用的请求超时的阈值设置,超时后 展示次数及平均延时 支持 url 重命名功能 支持自定义健康指数分值的比重
	组件监控	支持对云监控组件状态进行监控,宕机即告警,同时可下 载对应日志进行分析 支持对云监控部署主机进行监控,当主机出现异常时,进 行相关提示
	磁盘清理	支持对平台历史数据进行定时清理以及手动清理
	告警事件	支持查看每一条已发出的告警概要信息,可点击查看具体 告警
止故	告警规则	支持配置告警对象及通知方式、配置告警触发阈值、查看 并修改已有告警规则
	通知对象	支持新增修改查看相应告警发生所需要通知的单个对象
	通知对象组	支持新增修改查看相应告警发生所需要通知的对象组
网络监控	DNS 异常	支持对 DNS 请求异常多维度的统计的查询,结果以统计图 及列表展示
	事件检索	支持对集群事件进行分维度查询,以列表展示
	日志检索	支持对集群日志进行分维度查询 支持以关键字查询,结果高亮关键字
日志检索	资源检索	支持查询集群中 pod 资源,结果以列表展示
	操作审计	支持查询平台系统的一些关键操作记录
工具	TCP dump	支持对集群内的通信请求进行抓包,结果以文件形式保存 支持 arp,tcp,udp, lcmps 四种通信协议

2.4. 技术架构





云监控技术架构图

平台整体结构说明:

1.应用访问流量采集探针(hcmine):实时感知业务应用访问流量,业务响应质量(交易量,延时,错误率),最终实现业务关系调用分析和访问流量分析。

2.弹性 java 探针 (javaagent):通过应用不重启方式发现代码层面问题:实时感知系统中的代码异常,慢方法,外部服务调用。实时感知系统调用过程中数据库建联时间,SQL语句执行时间,数据库连接池。

3.海量信息处理层。将探针感知到的数据进行信息丰富,分类聚合,数据压缩入库。当有 弹性 java 探针的情况下信息处理层通过 traceid 进行数据关联,提供全链路信息的追踪,故障 快速定位,异常问题全息排查,多维度、全方位还原错误问题现场。

第 16 页



4. 可视化展示层。提供面向业务和运维场景的可视化分析能力。

2.5. 部署架构图

应用访问交互数据采集器 Hcmine 采用 daemonset 部署方式,以独立于业务容器的方式允许在容器云主机,对业务 pod 无侵入,探针资源消耗通过容器技术进行有效隔离。



平台高可用组件部署架构图

2.6. 案例展示

序号	类型	案例名称	集群规模	部署环境	当前状态	已上线主要应用 及应用类型
----	----	------	------	------	------	------------------



1.	能源	国网	客户容器集群规模,服务达 600 多个, pod3000 多个, 云主机 20 多台	物理机	测试环境 使用 上线准备 中	国网核心业务
2.	金融	浦发银行	容器 pod 达 5000,云主机 达 400 多	物理机	测试环境	互联网业务



第3章 云监控产品功能介绍



云监控平台模块架构图

云监控平台一方面实现了对云上应用整体进行宏观监控,提供了业务应用之间故障的定界 手段,另一方面通过关联业务指标及平台指标,实现了平台指标与业务指标的定界。通过对不 同纬度、不同来源的监控数据进行智能检索、模块化组装,提供出了整套可视化,易操作,易 分析的监控数据云平台。

- ▶ 通过总览,实现"集群一体化",对不同集群的重要指标数据进行统一展示;
- ▶ 通过资源监控,对多集群进行资源监控;
- ▶ 通过业务,以业务视角对云上运行应用整体宏观的监控;
- ▶ 通过应用,实现对应用性能代码级监控;
- ▶ 通过告警,对多集群异常指标率进行筛选,及时通知;
- ▶ 通过仪表盘,对集群内容器进行监控,图形化展示容器指标;

第 19 页



- ▶ 通过网络,对网络请求DNS 异常进行监控;
- ▶ 通过检索,对集群内日志信息进行智能检索,提供保留历史现场的能力;
- ▶ 通过工具,对集群内网络请求进行抓包,通过报文级别的网络异常信息;
- ▶ 通过配置,对平台及探针进行参数设置,实现数据动态配置。

3.1. 总览

对多集群目前调用情况及网络流量的监控,通过流量监控,了解集群及业务目前运行情况 及资源占用情况。可以对业务运行所需资源进行监控,做到及时调整和处理。

平台可以查看业务目前运行情况及资源占用情况,进行整体概览查看,对使用情况、运行状态进行评估。



如图所示,总览模块包括4个部分:业务概览、新增应用列表、集群概览、告警概览。

3.1.1. 业务概览



业务概览

查看更多>>

业务概览实现了对业务及其指标的统一归总展示,指标包括了业务的健康评分、访问次数 及错误率。平台根据智能算法计算出业务的健康程度,健康程度越低则排在最前列,使用户可 以第一时间了解所有业务的健康状况,从而对相关指标反映出的异常进行排查与处理。点击查 看更多跳转至业务列表,从而获取业务更详细的指标数据。如下图所示:

69业务 ◎ 102295 △ 0.1%	System 76 ◎ 106801 ▲ 0.1%	测试业务 ◎ 159850 ▲ 100%
OCR ⊚ 159850 ▲ 100%	云上监控 ◎ 9076 ▲ 0.15%	Default 87 ⊚ 0 ▲ 0%
bookdemo © 2349	服务测试 93 0.27%	hcmine
语言识别 ③ 2299	ctsec 100 ◎ 25 △ 0%	虚机业务测试 ◎ 2063

3.1.2. 新增应用列表

集群内应用的更替是频繁且往往是弱感知的,平台将检测到的新增应用汇总与此模块,以便用户及时了解集群内新增的应用,进而对相关应用进行业务配置,方便快捷。

新增应用列表展示了应用名称、应用所属集群、如已经加入业务,则显示所属业务、业务 的运行时间及一键加入业务配置操作。如下图所示:

第 21 页



新增应用列表

最近1天∨

名称	所在集群	所属业务	运行时间	操作
kafka	k8-68-dev		3小时	配置
10.10.102.93:31888	k8-68-dev		4小时	配置
10.10.101.73:31659	k8-68-dev		4小时	配置
10.10.102.93:31659	k8-68-dev		4小时	配置
10.10.102.92:30701	k8-68-dev		4小时	配置
10.10.102.92:31888	k8-68-dev		4小时	配置
10.10.101.73:31888	k8-68-dev		4小时	配置
10.10.101.73:30701	k8-68-dev		4小时	配置
10.10.102.93:30701	k8-68-dev		4小时	配置

3.1.3. 集群概览

集群的状态在一定程度上反应了集群内应用可否提供出正常的功能。集群概览对所监控的集群状态及资源使用情况进行了汇总展示,集群相关重要指标一目了然。

集群概览以集群的类型进行了2个纬度的划分:容器云、主机云。其展示了集群的多个指标信息:集群名称、集群状态、类型、CPU、内存、磁盘、节点数、服务数、pod数及工作负载数。如下图所示:



 集群概览							查看更多>
容器云 主机云							
名称	状态 🕐 🍷	类型 🔻	CPU 💠	内存 🛊	磁盘 💲	CPU利用率 🝦	内存使用率
k8-68-dev	●故障	裸机集群	60核	116G	875.82G	● 11.9核 19.84%	55.4G 47.
k8-159-rel	●正常	裸机集群	12核	22.92G	140.89G	(0.57核 4.73%	8.81G 38.
vm-50-test	●异常	虚拟集群					
k8-59-dev	• 正常	裸机集群	24核			[

3.1.4. 告警概览

告警数据的合理归并展示,最新告警的第一时间获知,历史告警的宏观分布情况等方面对于应相关人员维护集群正常运行往往是必须且有帮助的。

告警概览从三个方向统一展示了集群的告警信息:

未处理告警:展示从告警产生到目前位置还未处理告警数量的总数

告警等级分布图:展示告警等级的分布情况

高等级告警信息:展示高等级的告警信息

如下图所示:



告警概览				查看更多>>
④ 当前未处理告警				1041
告警等级分布图 1,200 1,000 800 600 400 200 0				
P1 process-exporter-bb8 k8-68-dev 集群下monitorin 1.0,当前值3.18	P2 rd P1 Podf ng命名空间下proce	P3 告警 ess-exporter-bb8r	P4 2021-12-2 dpod cpu使用率(9	P5 8 17:00:02 6) 每次大于
process-exporter-cqh k8-68-dev 集群下monitori 1.0,当前值3.77	p6 P1 Pod ng命名空间下proce	告警 ess-exporter-cqhp	2021-12-2 6pod cpu使用率(⁶	8 17:00:02 %) 每次大于
testdemo1-74457455	58-thswc	Pod告警	2021-12-2	8 17:00:02

3.2. 资源监控

在集群的一系列监控指标中,资源监控指标的重要程度可以说是数一数二的。资源的不足,资源分配的不合理对于整个集群的正常工作都会巨大影响。



在资源监控中,划分出以 k8s 集群资源监控、主机集群资源监控、资源列表、3D 机房 4 个模块为主的具体监控面。

3.2.1. 机房资源监控

通过抽象机房机柜坐标位置进行 3D 建模,构建出与实际机房相对应的 3D 机房,便于从宏观纬度查看机房服务器的状态及资源使用情况。

3D 机房模块从机房机柜服务器三个纬度,清晰地展示了机房服务器的可视化图像。

3.2.1.1. 机房

从宏观纬度展示了机房内机柜的分布情况,可通过放大缩小及旋转等方式对整个机房进行 多角度观察。如下图所示:



3.2.1.2. 机柜

点击进入机柜层面,可以清楚的看到机柜里面服务的数量及状态:红色代表服务器有告警或者 服务器本身有异常。





3.2.1.3. 服务器

点击服务器进入具体服务器层面,可查看单台服务器基本具体信息:

服务器基本信息:展示服务器的基本信息

正在运行的 pod: 展示此台服务器内正在运行的 pod

未处理的告警:展示此台服务器未处理的告警

切换成图表界面,可查看服务 CPU、内存、磁盘使用情况。

第 26 页



如下图所示:

基本信息 图表信息	X 查看详情>>
	集群名称:k8–68–dev
状态:异常	开始槽位:3
上架时间:2021–11–10 10:32:21	服务器高度: 1U
所属机柜:测试机柜	型号:
序列号:	供应商:
负责人:	CPU: 4核
GPU: 0核	内存: 7.64G
磁盘: 94.06G	操作系统:Linux
IP地址:10.10.102.93	
正在运行POD(9) node-exporter-6gpnn process-exporter-gsr	ısm apm-abnormal-alarm-0
apm-es-server-0 metrics-server-7c55c9fbc	b-9t8vt calico-node-6j7s6
hcmine-agent-flmf9 kube-proxy-bxbc5 a	lertmanager-main-0
未处理告警(84)	
k8-68-dev 隼群下vvzl命名空间下apm-abnormal-	-alarm-0pod cpu使用率(%) 每次大干1.0.当前值3.78





3.2.2. 集群资源监控

在集群运行之时,期望能方便查看集群相关的一些指标及状态,进而判断集群是否健康。 由此,平台提供了集群资源监控模块统一对多集群数据进行展示。

如在集群概览层面,可查看多集群资源分配情况,判断集群状态情况及资源使用情况,从 CPU、内存、告警等多方面确认集群健康状态。

平台对不同类型的集群进行了划分,分为 K8s 集群及主机集群。

3.2.2.1. K8s 集群监控



3.2.2.1.1. 集群列表

选择进入 k8s 集群 tab 页面,可以看到 k8s 集群列表, k8s 集群的大部分重要信息:集群的 名称、数据中心、状态、类型、CPU、内存、磁盘、节点、服务、pod 工作负载等信息在列表 内一目了然。如下图:

集群名称: 输入集群名	名称搜索	数据中	叩心: 全部		\vee				I	查询
导入集群新	曾集群									
名称	数据中心	状态 ⑦ 🏾	类型 🔻	CPU 💲	内存 💲	磁盘 💲	CPU利用率 🍦	内存使用率 💲	节点数 💲	操作
k8-68-dev	上海数据 中心	●故障	裸机集群	60核	116G	875.82G	● 8.11核 13.51%	57.07G 49.2%	9	编辑 删除
k8-159-rel	杭州数据 中心	 故障 	裸机集群	12核	22.92G	140.89G	(0.56核 4.68%	8.72G 38.03%	3	编辑 删除
vm-50-test	杭州数据 中心	• 故障	虚拟集群						3	编辑 <mark>删除</mark>
k8-59-dev	杭州数据 中心	•故障	裸机集群	24核					6	编辑 删除

集群列表也提供了集群新增编辑删除功能,因与集群新增类似故此处对于编辑功能省略, 具体可参考集群新增模块。如下图:



新建集群		×
* 集群标识:		
*集群名称:		
*集群类型:	V	
数据中心:	\vee	
Prom访问地址:	测 试	
ILB IP:		
SLB IP:		
VIP:		
容器集群api配置	测 试	
* 认证方式:	Token认证 V	
* api_url地址:		

输入集群的一些相关信息之后保存便可新建一个集群。

列表中选择一个集群点击进入即可跳转到某个集群详情页面,多集群进入单集群节点后判断集群状态情况及资源使用情况、可查看集群、节点数据、资源使用情况等详细数据。



3.2.2.1.2. 概览

包括了集群的 CPU、内存、磁盘当前使用值及 pod 启动率,并用图表的方式展示了 CPU、 内存、磁盘、GPU 等相关资源的历史使用情况。如下图:

概览	详情	告警(159)	节点(9)	命名空间(17)	服务(45)	工作负载(58)	Pod(99)	管理组件	
CPU利用率 18.33%		CPU Request 35.62核	CPU Limit 39.65核	内存利用率 48.98%	内存 Request 62.11G	内存 Limit 73.26G	磁盘和 44.7	利用率 Po 7 6% 96	^{出启动率} .97%
CPU使用率			1	内存使用率		磁	盘使用率		
21% 18% 15% 9% 6% 3% 0% 12~21 14:55	Andolly 5 12-27	22:10 12-22 05:25	12-22 12:40	70% 60% 50% 40% 30% 20% 10% 0% 12-21 14:55 12-21 22	10 12-22 05:25 1	MLM 2-22 12:40	60% 50% 40% 20% 10% 0% 0%	21 22:10 12-22 05:25	12-22 12:40
CPU配额				内存配额		GP	U使用率		
 Allocatable 60核 0 50核 40核 0 30核 	e 🔳 Reque	est Limit 0		Allocatable Request 130.396 111.766 93.136 74.516	Limit				

3.2.2.1.3. 详情

展示集群的基本信息

概览	详情	告警(999+)	节点(9)	命名空间(18)	服务(52)	工作负载(60)	Pod(115)	管理组件
集群版本								
kubernetes版	本:v1.18.20		控制节点IP:-					
访问地址								
ILBIP:-			SLBIP:-			VIP:-		
组件访问地址	Ŀ							
prom地址:10.	10.101.69:300	91						



3.2.2.1.4. 告警

概览	详情	告警(159)	节点(9)	命名空间(17	7)	服务(45)	工作负载(58)	Pod(99)	管理组件		
								请输入	告警内容搜索		٩
告警内容		告警对象		告警类型	告警 等级	≑ 通知对象		告警时间 💠	状态 👻	操作	
statefulset:	alertmanager.	alertmanag	er-main	工作负载告警	P3	윦 Jonny 윦 Gyro		2021-12-22 10:38:00	◎ 未处理	处理	
statefulset:	alertmanager.	alertmanag	er-main	工作负载告警	P3	ጼ Jonny ጼ Gyro		2021-12-22 10:38:00	◎ 未处理	处理	
statefulset:	alertmanager.	alertmanag	er-main	工作负载告警	P3	ጼ Jonny ጼ Gyro		2021-12-22 10:38:00	◎ 未处理	处理	
statefulset:	prometheus	prometheus	s-k8s	工作负载告警	P3	용 Jonny 용 Gyro		2021-12-22 10:38:00	◎ 未处理	处理	
statefulset:	prometheus	prometheus	s-k8s	工作负载告警	P3	유 Jonny 유 Gyro		2021-12-22 10:38:00	◎ 未处理	处理	

3.2.2.1.5. 主机

切换主机页,可查看此集群相关的主机列表。

关于主机监控,可见主机资源监控章节,此处不做详述。

3.2.2.1.6. 服务

切换服务页,可查看此集群相关的服务列表。

关于服务监控,可见服务监控章节,此处不做详述。

3.2.2.1.7. 工作负载

切换工作负载页,可查看此集群相关的工作负载列表。

关于工作负载监控,可见工作负载监控章节,此处不做详述。



3.2.2.1.8. Pod

切换 Pod 页, 可查看此集群相关的 pod 列表。

关于 pod 监控, 可见 pod 监控章节, 此处不做详述。

3.2.2.1.9. 管理组件

展示 kube-system 命名空间下的 pod 信息

概览	详情	告警(999+)	节点(9)	命名空间(18)	服务(52)	工作负载(60)	Pod(115)	管理组件	
							输入关	建词搜索	Q
名称				状态 ② <table-cell></table-cell>	节点名和	称		节点IP	
calico-kub	e-controllers	-59877c7fb4-wtk57		• 正常	10.10.7	101.68-master		10.10.101.68	
calico-nod	le-2xzh8			• 正常	10.10.7	101.71-database-2		10.10.101.71	
calico-nod	le-6j7s6			• 正常	10.10.1	102.93-salve		10.10.102.93	
calico-nod	le-dl5rw			• 正常	10.10.7	101.72-database-3		10.10.101.72	
calico-nod	le-gqkw4			• 正常	10.10.7	102.91-salve		10.10.102.91	
calico-nod	le-jh7qx			• 正常	10.10.1	101.73-prometheus-da	ta	10.10.101.73	

3.2.2.2. 主机集群资源监控

3.2.2.2.1. 主机集群列表

选择进入主机集群 tab 页面,可以看到主机集群列表,列表内容涵盖了主机集群的大部分重要信息:集群的名称、数据中心、状态、类型、CPU、内存、磁盘、节点等信息。如下图:

名称	数据中心	状态 💿 🍷	类型	CPU 🌲	内存 🌲	磁盘 🝦	CPU利用率 🍦	内存使用率 💲	节点数	操作
155集群	杭州数据 中心	• 正常	虚拟机集群	12核	22.92G	140.89G	4.03核 33.62%	5.67G 24.72%	3	编辑删除
测试用虚机集群		●异常	物理机集群						2	编辑 删除



与 k8s 集群相比, 主机集群的类型为虚拟机集群, 且少了命名空间, pod 等 k8s 集群特有的资源。

选择列表内某个主机集群,可跳转至单个集群详情页面,其包含了概览,告警,节点三个 tab 页。

3.2.2.2.2. 概览

展示了主机集群的 CPU、内存、磁盘使用率的当前数值及历史折线图。

状态 ⑦: 正常 部署地: 杭州	集群标识: zanebono_test 数据中心: 杭州数据中心	类型: 虚拟机集群
概览 告警(1) 节点(3)		
CPU利用率 33.62%	内存利用率 24.74%	磁盘利用率 11.78%
CPU使用率	内存使用率	磁盘使用率
35% 30% 25% 20% 15% 10% 5% 0% 12-29 16:53 12-29 17:13 12-29 17:14 12-29 17:14 12-29 17:14 12-29 17:14 12-29 17:14 12-29	25% • • • • • • • • • • • • • • • • • • •	12%

3.2.2.2.3. 告警

展示主机集群相关的告警信息

					请输入告	警内容搜索(2
告警内容	告警对象	告警类型	告警 等级	通知对象	告警时间 💠	状态 💿 操作	
vm: 10.10.102.155:9090	10.10.102.155:9090	应用告警	P1	R testGroup1	2021-12-28 23:40:00	• 已处理 已处理	
					第 1-1 条/总共 1条	< 1 > 10条/页 >	•



3.2.2.2.4.节点

切换主机页,可查看此主机集群相关的主机列表。

关于主机监控,可见主机资源监控章节,此处不做详述。

3.2.3. 主机资源监控

当主机资源紧张时,期望能便捷地查看多台主机状态情况及资源使用情况。平台提供了主机健康模块,对主机状态及资源数据进行汇总展示。平台对与不同类型集群的主机进行了划分: k8s集群及主机集群。

3.2.3.1. 主机列表

在主机列表页面,可以查看到主机层面的 CPU、内存、磁盘等资源使用情况。

✤ K8s 集群:

K8s 主机列表展示了 k8s 主机相关的重要指标,包括了主机名称、主机状态、CPU、内存、磁盘、IP、是否污点及运行业务。如下图:



云监控(HC-CloudMonitor)产品说明书

名称	状态 💿 🍷	CPU 🌲	内存 💲	磁盘 🗅	CPU利用率 🍦	内存使用率 🗅	IP	污点情况	运行业务
10.10.101.69- gateway	● 异常	8核	15.51G	98.94G	5.49核 68.57%	10.21G 65.8%	10.10.101.69	否	
10.10.102.92- salve	●故障	4核	7.64G	94.06G	1.4核 35.07%	4.56G 59.72%	10.10.102.92	否	
10.10.101.71- database-2	• 异常	8核	15.51G	98.94G	2.63核 32.87%	8.84G 57%	10.10.101.71	否	
10.10.101.73- prometheus-data	• 异常	8核	15.51G	98.94G	2.01核 25.16%	7.51G 48.38%	10.10.101.73	否	
10.10.102.93- salve	• 异常	4核	7.64G	94.06G		3.34G 43.68%	10.10.102.93	否	
10.10.101.70- database-1	• 异常	8核	15.51G	98.94G		10.36G 66.78%	10.10.101.70	否	
10.10.102.91- salve	• 异常	4核	7.64G	94.06G	●0.74核 18.56%	3.43G 44.87%	10.10.102.91	否	

◆ 主机集群:

主机集群列表展示了主机集群的主机相关的重要指标,包括了主机名称、主机状态、CPU、 内存、磁盘、主机 IP、宿主机 IP 及运行业务。并提供了对主机的移除功能。

名称	内存 🌲	磁盘 💲	CPU利用率 🍦	内存使用率 👙	主机IP	宿主机IP	运行业务	操作
10.10.102.155	7.64G	46.96G	4 核 99.93%	2.69G 35.18%	10.10.102.155		虚机业务测试 testdemo测试	移除
10.10.102.156	7.64G	46.96G	0.02核 0.51%	• 1.49G 19.5%	10.10.102.156		虚机业务测试 testdemo测试	移除
10.10.102.157	7.64G	46.96G	0.02核 0.43%	- 1.5G 19.68%	10.10.102.157		虚机业务测试	移除

从主机列表层面出发,进入单台主机资源层面,可查看单台主机的概览、此台主机的相关 告警信息等信息。

3.2.3.2. 概览

✤ K8s


K8s 主机概览展示了 k8s 主机相关的重要指标,包括了 CPU、内存、磁盘、GPU 的使用情况。



◆ 主机集群

主机集群主机概览展示了主机相关的重要指标,包括了 CPU、内存、磁盘的使用情况



3.2.3.3. 详情



平台只展示了 k8s 集群的详情, 主机集群无需展示。

概览 详情 告警(151) POD(9)	
基本信息 资产信息 标签 注释	
标签名称	标签值
beta.kubernetes.io/arch	amd64
beta.kubernetes.io/os	linux
kubernetes.io/arch	amd64
kubernetes.io/hostname	10.10.102.92-salve
kubernetes.io/os	linux
yyzl	ctsec
	第 1-6 条/总共 6条 < 1 > 10 条/页>

3.2.3.4. 告警

展示主机相关的告警信息

概览 告警(1) 节点(3)							
					请输入告警	内容搜索	Q
告鑒内容	告警对象	告警类型	告警 等级	通知对象	告誓时间 💠	状态 😨	操作
vm: 10.10.102.155:9090: 连接失败次数:当前监控值21。触发	10.10.102.155:9090	应用告警	P1	의 testGroup1	2021-12-28 23:40:00	 已处理 	

3.2.3.5. Pod

切换 Pod 页,可查看此集群相关的 pod 列表。

关于 pod 监控, 可见 pod 监控章节, 此处不做详述。

3.2.3.6. 进程

展示主机集群主机的进程信息



概览 告警(0)	进程			
			输入关键词搜索	Q,
名称	占用端口	CPU利用率 🖕	内存使用率 👙	启动路径
process-exporte	9256	0.02 0.57%	51.22M 0.65%	
prometheus	9090	0.02 0.5%	• 404.68M 5.17%	
hcmine_go	13133,8888,1777,55679	0.004 0.1%	34.12M 0.44%	
java	9191	7.0E-4 0.02%	8 91.98M 11.4%	
test-netserver	9200	6.0E-4 0.02%	12.65M 0.16%	

3.2.4. 命名空间监控

为了快速掌握命名空间层面的整体资源及运行状态,平台提供了一个统一的模块页面展示 各个命名空间的状态及资源使用情况。

3.2.4.1. 命名空间列表

如期望查看各个命名空间的资源使用情况,可以从命名空间列表层面中看到各个命名空间的状态、CPU、内存、磁盘使用和运行情况及命名空间相关的 pod 及工作负载数量关系信息如下图:



云监控(HC-CloudMonitor)产品说明书

名称	状态 💿 🍷	CPU使用率 🍦	内存使用率 🝦	Pod启动率 🍦	服务数 💲	Pod数 🝦	Deployment	工作负载数 Damonset	Statefulset
bookdemo	• 正常	0核 0%	0%	0 %	1	1	1	0	0
cattle-system	• 正常	0.02核 0%	522.21M 0%	100 %	0	1	1	0	0
cloudmonitor	● 故障	4.43核 21.1%	30.18G 63.91%	100 %	20	22	4	0	14
default	• 正常	• 0.01核 5%	0%	100 %	1	1	1	0	0
fleet-system	● 故障	0.01核 0%	55.15M 0%	100 %	0	1	1	0	0
hcmine	● 故障	7.17核 0%	2.9G 123.71%	88.89 %	0	9	1	1	0
ingress-nginx	● 故障	5.38核 53800%	2.826 14424.28%	66.67 %	1	3	2	0	0
kube-node-lease	• 正常	0%	0%	0 %	0	0	0	0	0
kube-public	• 正常	0%	0%	0 %	0	0	0	0	0

3.2.4.2. 概览

从命名空间资源列表层面出发,进入单个命名空间资源概览层面,可查看命名空间的 CPU、 内存、磁盘、GPU 的当前及历史使用情况及当前命名空间下的 pod 启动率。



3.2.4.3. 详情



切换 tab 页转换到命名空间详情页,可查看此命名空间的 yaml 信息:

metadata:	
annotations:	
cattle.io/status: '{"Conditions":[{"Type":"ResourceQuotaInit","Status":"True","Message":"","LastUpdateTime":"202	-11-29T03:53:45Z"},{"Type":"InitialRolesPopulated","Status":"True","Message":"","LastUpdateTime":"2021-11-29T03:53:45Z"}}}'
field.cattle.io/projectld: c-wt49t:p-shdmd	
lifecycle.cattle.io/create.namespace-auth: 'true'	
creationTimestamp: '2021-11-29T11:44:51.000+08:00'	
finalizers:	
- controller.cattle.io/namespace-auth	
labels:	
field.cattle.io/projectld: p-shdmd	
managedFields:	
- apiVersion: v1	
fieldsType: FieldsV1	
fieldsV1:	
f:status:	
f:phase: {}	
manager: kubectl	
operation: Update	
time: '2021-11-29T11:44:51.000+08:00'	
- apiVersion: v1	
fieldsType: FieldsV1	

3.2.4.4. 告警

切换到告警页,可以查看此命名空间纬度下的告警信息列表:

告警内容	告警对象	告警类型	告警等级 💠	所属业务	通知对象	告警时间 ≑	状态 🛒	操作
k8-68-dev 集群下cloudmonitor命名空	aerospike-0	Pod告警	P1		RDIO	2021-12-28 17:00:00	• 已处理	已处理
k8-68-dev 集群下cloudmonitor命名空	es-cluster-0	Pod告譬	P1		RDIO	2021-12-28 17:00:00	 已处理 	已处理
k8-68-dev 集群下cloudmonitor命名空	nephele-0	Pod告譬	P1		RDIO	2021-12-28 17:00:00	• 已处理	已处理
k8-68-dev 集群下cloudmonitor命名空	apm-data-analyzer-0	Pod告警	P1		RIDIO	2021-12-28 16:59:00	。 未处理	处理
k8-68-dev 集群下cloudmonitor命名空	apm-es-server-0	Pod告警	P1		RDIO	2021-12-28 16:59:00	。 未处理	处理
k8-68-dev 集群下cloudmonitor命名空	apm-data-receiver-0	Pod告警	P1		RDIO	2021-12-28 16:59:00	。未处理	处理

3.2.4.5. 服务

切换到服务页,可查看此命名空间相关的服务列表。

关于服务监控,可见服务监控章节,此处不做详述。

3.2.4.6. 工作负载

切换到工作负载页,可查看此命名空间相关的工作负载列表。

第 41 页



关于工作负载监控,可见工作负载监控章节,此处不做详述。

3.2.4.7. pod



3.2.5. 服务监控

当服务创建之后,期望能方便快速的查看服务及相关端点等信息。由此平台提供了一个服务模块统一提供服务相关信息数据。

3.2.5.1. 服务列表

服务列表内容包含了服务名称、服务相关命名空间、服务类型及相关 IP 等指标,=,由此可初步判断服务的运行是否正常及相关端点数是否符合预期:



云监控(HC-CloudMonitor)产品说明书

概览	详情	告警(999+)	节点(9)	命名空间(18)	服务(52)	工作负载(60)	Pod(115)	管理组件			
										51	1入关键词搜索 Q.
名称			命:	名空间	类型 👻	服务地址			端点数 💠	关联POD数 👙	运行时间 💠
bookdem	o l		bo	okdemo	ClusterIP	10.100.90.106	8080		1	1	0月30天
aerospike	-svc		clo	udmonitor	ClusterIP	10.110.194.27	53000,3002		2	1	0月30天
apm-abno	ormal-alarm-sv	rc -	clo	udmonitor	ClusterIP	None:			0	0	0月30天
apm-conf	igserver-svc		clo	udmonitor	NodePort	10.97.182.254	8888,6659		2	1	0月30天
apm-data	-analyzer-svc		clo	udmonitor	ClusterIP	None:			0	0	0月30天
apm-data	-receiver-svc		clo	udmonitor	NodePort	10.108.223.19	3:57001,7001,808	D	3	1	0月30天
apm-es-s	erver-svc		cla	udmonitor	ClusterIP	10.106.178.68	9191		1	1	0月30天
apm-pron	netheus-webho	ook-svc	clo	udmonitor	ClusterIP	10.102.72.241	23333		0	0	0月30天
applicatio	n-monitor-svc		clo	udmonitor	ClusterIP	10.104.28.219	9090		1	1	0月30天
elasticsea	rch-svc		clo	udmonitor	ClusterIP	10.101.78.86:9	200,9300		6	3	0月30天

3.2.5.2. 概览

在服务概览页面,可以看到多个服务运行情况以及查看服务相关端点数量信息如下图:

类型:Clus 运行时间:	sterIP 23天		集群IP:10.7	101.78.86		外部IP:				
概览	yaml信息	告警(0)	工作负载(1)	POD(3)	分层拓扑					
服务地址										
10.101.78 端点(end	10.101.78.86:9200,9300 端点(endpoints)									
地址			端口	目标的	陸型	协议 👅	目标			
192.168	3.0.110		9200	POD		ТСР	es-cluster-0			
192.168	8.0.110		9300	POD		ТСР	es-cluster-0			
192.16	3.77.145		9200	POD		ТСР	es-cluster-2			
192.168	8.77.145		9300	POD		ТСР	es-cluster-2			
192.168	3.240.114		9200	POD		ТСР	es-cluster-1			

3.2.5.3. yaml 信息



跳转 yaml 信息页, 可查看此服务相关的 yaml 信息, 包括标签、注释、yaml 文件内容, 如

下图:

既览	yaml信息	告警(0)	工作负载(1)	POD(1)	分层拓扑	
示签	主释 yaml					
metadata: creation ¹ managed - apiVers fieldsTy fieldsV fieldsV	: Timestamp: '202' dFields: sion: v1 rpe: FieldsV1 1: :	I-11-29T11:44:52	2.000+08:00'			
f:por .: {} k:{"	ts: port":3002,"proto	ocol":"TCP"}:				
.: { f:n f:p	} ame: {} iort: {}					
f:ta k:{"	argetPort: {} port":53000,"pro	tocol":"TCP"}:				
f:n f:p f:p	ame: {} ort: {} irotocol: {}					
f:ta f:sele	argetPort: {} ector:					
f:na f:ses	me: {} sionAffinity: {}					
f:type manage	e: {} er: kubectl					

3.2.5.4. 告警

展示与此服务相关的告警信息

3.2.5.5. 工作负载

切换工作负载页,可查看此服务相关的工作负载列表。

关于工作负载监控,可见工作负载监控章节,此处不做详述。

3.2.5.6. Pod

切换 Pod 页,可查看此服务相关的 pod 列表。

第 44 页



关于 pod 监控, 可见 pod 监控章节, 此处不做详述。

3.2.5.7. 分层拓扑

展示与此服务相关的分层拓扑,从命名空间->服务 -> 工作负载 -> 节点->POD -> 容器等纬度的数量、名称及状态。支持回溯功能,如下图:

概览	yaml信息	告警(0)	工作负载(1)	POD(1)	分层拓扑			
命名空间	1						۵	溯
1	2					cloudmonitor		
服务 1						co aerospike-svc		
1	2					日 aerospike 道利章		
节点 1						E 10.10.101.73		
Pod 1						eerospike-0		
 1						(®) aerospike		

点击各个纬度的节点可查看各个纬度的详细信息:

♦ 命名空间详情:

cloudmonitor		查看详情	×
详情			
状态:正常	服务数: 22		
Pod数: 19	Deployment: 4		
DaemonSet: 0	StatefulSet: 14		



◆ 服务详情

aerospike-svc	
---------------	--

详情

类型: ClusterIP 关联Pod数: 1 运行时间: 0月30天 所属租户: cloudmonitor 端点数: 2 所属命名空间: cloudmonitor

◆ 工作负载详情



aerospike

查看详情 X

详情

期望实例个数: 1
创建时间: 2021-06-07 15:28:13
所属租户: cloudmonitor
运行时间: 0月30天

所属命名空间: cloudmonitor

yaml

apiVersion: apps/v1 kind: StatefulSet metadata: creationTimestamp: '2021-11-29T11:44:56.000+08:00' generation: 3 managedFields: - apiVersion: apps/v1 fieldsType: FieldsV1 fieldsV1: f:spec: f:podManagementPolicy: {} f:replicas: {} f:revisionHistoryLimit: {} f:selector: f:matchLabels: .: {} f:name: {} f:serviceName: {} f:template: f:metadata: f:labels: .: {} f:name: {} f:spec: f:containers: k:{"name":"aerospike"}: .: {} f:image: {} f:imagePullPolicy: {} f:name: {} finarta



◆ 节点详情

详情

状态:正常

允许的Pod数量: 110

系统版本: 11197179

IP: 10.10.101.73

CPU核数: 8核

内存: 15.51G

磁盘: 98.94G

CPU使用量



内存使用量





✤ pod 详情

aerospike-0		查看详情 X
详情		
IP: 192.168.38.152 所属租户: cloudmonitor 所属命名空间: cloudmonitor	阶段: Running 所属工作负载: aerospike	
告警内容	告警类型 😨 告警时间	告警状态
	暂无数据	

◆ 容器详情



apm-es-server

查看详情

详情

状态:正常

ContainerID: docker://9fa283778b3c0246b6d0bf53f1b1e79efd534dedd2bb9fc0545ec52b1f12adc2

运行的Pod名称: apm-es-server-0

运行的Namespace: cloudmonitor

镜像ID: docker-pullable://10.1.11.205/k8s-deploy-test/apm-es-

server@sha256:440672e93fe3d4b79479439c4f1ee37204e468a68f6a805a72f78be588ef585a

镜像名称: 10.1.11.205/k8s-deploy-test/apm-es-server:v1.0.0

配置CPU信息: 1B

配置内存信息: 2G

配置磁盘信息: 97.95G

CPU使用量









3.2.6. 工作负载监控

当工作负载工作之后,期望能准确快速的查看工作负载相关状态及资源情况。平台提供了 工作负载模块页面,对工作负载数据进行汇总展示,方便当工作负载发生异常时对异常问题进 行快速排查。

3.2.6.1. 工作负载列表

跳转只工作负载 tab 页,可查看工作负载相关的指标。平台统计常见的三类主要类型的工作负载:deployment、daemonset、statefulset。工作负载列列表指标涵盖了工作负载名称、工作负载状态、相关的命名空间、相关期望 pod 数与实际可用 pod 数及 CPU、内存的 request 值。如下图:

概览 详情 告警(15	59) 节点	(9) 命名雪	2间(17) 服务	务(45) 工作负载(58)	Pod(100) 1	管理组件
deployment daemonset	statefulset				请输入名称挑	l索 Q
名称	状态 🕐 🍷	命名空间	Pod数 🌲	CPU Request 🍦	内存 Request 🌲	运行时间 🝦
cattle-cluster-agent	• 正常	cattle- system	山 1 可用:1	0核	0	23天
event-exporter	• 正常	cloudmonitor	山 1 可用:1	0.2核	200M	23天
grafana	• 正常	cloudmonitor	山 1 可用:1	0.2核	200M	23天
default-http-backend	 故障 	ingress- nginx	山 1 可用:1	0.01核	20M	23天
nginx-ingress-contro	● 正常	ingress- nginx	山 1 可用:1	0核	0	23天
calico-kube-controll	• 正常	kube-system	山 1 可用:1	0核	0	23天
coredns	• 正常	kube-system	山 2 可用:2	0.2核	140M	23天
kube-state-metrics	 故障 	monitoring	山 1 可用:1	0核	0	23天



在工作负载概览层面,可以查看多个工作负载的类型以及相关资源申请的情况,由此判断 工作负载状态情况和资源使用情况,从期望 pod 及实际运行 pod 数量、告警等多方面确认工作 负载健康状态。

3.2.6.2. 概览

进入某个工作负载概览层面,通过查看单个工作负载的 CPU、内存、磁盘、GPU、pod 启动 率等更加详细的使用情况及此个工作负载的相关告警信息。



3.2.6.3. yaml 信息

切换到 yaml 信息页面,可查看此工作负载相关的 yaml 信息,包括标签、注释、yaml 文件内容,如下图:



概觉 yaml信息 告警(22) 服务(1) Pod(1)	
标签 注释 yaml	
注释名称	注释值
field.cattle.io/publicEndpoints	[{"addresses"
"port"	30001
"protocol"	"TCP"
"serviceName"	"cloudmonitor
"allNodes"	true}
{"addresses"	["10.10.101.69"]
"port"	30701
"protocol"	"TCP"
"serviceName"	"cloudmonitor
"allNodes"	true}

3.2.6.4. 告警

切换到告警页面,可查看此工作负载相关的告警信息

3.2.6.5. 服务

切换到服务页,可查看此工作负载相关的服务列表。

关于服务监控,可见服务监控章节,此处不做详述。

3.2.6.6. Pod

切换 Pod 页,可查看此工作负载相关的 pod 列表。

关于 pod 监控, 可见 pod 监控章节, 此处不做详述。

3.2.7. pod 监控



当业务发生异常时,第一步往往偏向查看相关 pod 的一些状态,由此初步判断异常的排查的切入方向。由此,平台提供了 pod 模块页面,对 pod 相关信息进行了汇总展示,方便对 pod 信息的快速准确的查看。

3.2.7.1. pod 列表

在 pod 列表层面,可查看多个 pod 的状态运行阶段以及相关容器情况进而判断 pod 健康程度。如下图:

概览 详情	告警(999+)	节点(9)	命名空(间(18) 服务(5	2) 工作负载(60)	Pod(116)	管理组件				
名称	IP	状态 ③ 🍵	阶段	命名空间	所属工作负载	容器数 💠	重启次数 💠	运行时间 💠	CPU Request 🍦	内存 Request 👙	CPU使用量
node-exporte	10.10.101.72	 故障 	Running	monitoring	node-exporte	ulu 2 可用:2	2	5天	0.11核	200M	
process-expo	10.10.102.92	 故障 	Running	monitoring	process-expo	al. 1 可用:1	9	30天	0.26核	50M	
log-data-par	192.168.175.1 42	 故障 	Running	yyzl	log-data-par	山 1 可用:1	0	16天	1核	2G	
prometheus-a	192.168.38.16 2	 正常 	Running	monitoring	prometheus-a	山 1 可用:1	9	30天	0核	0	
kube-proxy-n	10.10.101.72	 正常 	Running	kube-system	kube-proxy	al. 1 可用:1	16	30天	0核	0	
testy-pod	192.168.106.1 39	 正常 	Running	testns		山 1 可用:1	23	29天	0核	0	
hcmine-agent	10.10.101.68	 正常 	Running	hcmine	hcmine-agent	di. 1 可用:1	0	1天	0核	300M	
hcmine-agent	10.10.102.93	• 正常	Running	hcmine	hcmine-agent	di. 1 可用:1	0	8天	0核	300M	
kube-schedul	10.10.101.68	• 正常	Running	kube-system	10.10.101.68	al. 1 可用:1	40	30天	0.1核	0	
cattle-clust		• 异常	Failed	cattle-system	cattle-clust	ul. 1 可用:0	0	3天	0核	0	-

pod 列表页包括了 pod 名称、ip、状态、阶段、命名空间、所属工作负载、期望容器数、可用容器数、重启次数、运行时间、CPU request、内存 request、CPU 使用量、内存使用量等指标信息。

3.2.7.2. 概览

在 pod 概览层面,通过查看单个 pod 的 CPU、内存、磁盘、重启次数等更加详细的指标信息判断 pod 运行状态及资源使用情况。





3.2.7.3. 详情

可查看此 Pod 相关的 yaml 信息,包括标签、注释、yaml 文件内容,如下图:

概范 <mark>详情</mark> 告聲(30) 事件(0) 容器(2)	
标签 注释 Yaml	
标签名称	标签值
app.kubernetes.lo/name	node-exporter
app.kubernetes.lo/version	v1.0.1
controller-revision-hash	585c698895
pod-template-generation	1

3.2.7.4. 告警

切换到告警页面,可查看此 pod 相关的告警信息:



概览 详情 告警(30)	事件(0) 容器(2)						
告警内容	告讐对象	告警类型	告警等级 💲	所属业务	通知对象	告警时间 👙	状态 👘	操作
k8-68-dev 集群下monitoring命名空间…	node-exporter-8q2rz	Pod告警	P1	云上监控 69业务	RDIO	2021-12-28 17:00:00	• 已处理	
k8-68-dev 集群下monitoring命名空间	node-exporter-8q2rz	Pod告警	P1	云上监控 69业务	RIDIO	2021-12-28 16:57:00	◎ 未处理	处理
k8-68-dev 集群下monitoring命名空间	node-exporter-8q2rz	Pod告警	P1	云上监控 69业务	۶.DIO	2021-12-28 16:54:00	◎ 未处理	处理
k8-68-dev 集群下monitoring命名空间	node-exporter-8q2rz	Pod告警	P1	云上篮控 69业务	RDIO	2021-12-28 16:45:00	◎ 未处理	处理
k8-68-dev 集群下monitoring命名空间	node-exporter-8q2rz	Pod告譬	P1	云上监控 69业务	RIDIO	2021-12-28 16:31:00	◎ 未处理	处理
k8-68-dev 集群下monitoring命名空间	node-exporter-8q2rz	Pod告警	P1	云上监控 69业务	Pi DIO	2021-12-28 16:23:00	。 未处理	处理

3.2.7.5. 事件

切换到事件页面,可查看此 pod 相关的事件信息

3.2.7.6. 容器

切换容器页,可查看此 pod 相关的容器列表。

关于容器监控,可见容器监控章节,此处不做详述。

3.2.8. 容器监控

在业务不能满足预期功能时,往往需要确认相关容器是否工作正常。平台对容器数据进行 了汇总,方便对容器的运行状况及资源使用情况进行复核。更详细的容器指标可查看<u>仪表盘</u>模 块。

3.2.8.1. 容器列表

在容器列表里可用看到容器的一些指标,包括容器的名称、状态、命名空间、CPU使用量 及内存使用量。



概览	详情	告警(0)	事件(1)	容器(1)						
									请输入名称	Q
名称		状态		🔹 命名空间]	CPU	J使用量 👙		内存使用量 ≑	
calico-node		Running	9	kube-sy	stem	0.0	3核		30.49M	
								第 1-1 条/总	共 1条 < 1 >	10 条/页 ∨

3.2.8.2. 概览

容器概览层面,可通过查看容器的状态运行阶段来判断容器是否正常,可通过查看 CPU, 内存的使用量来判断容器运行是否符合期望。

阶段:Running 运行时间:2天	主机IP: 10.10.101.69	Pod IP: 10.10	101.69
概览 详情			
CPU使用量 0.04核	内存使用量 30.22M	磁盘使用量 31.3M	重启次数 26
CPU使用量 0.08核 0.07核 2021-12-21 17:10:08	内存使用量 38.15M	磁盘使 33.38M 	
0.05核 0.04核 0.04核 0.03核 0.02核 0.01核	9.54M	23.84M - 19.07M - 14.31M - 9.54M - 4.77M -	
0核 12-21 17:10 12-22 01:35 12-22 10:00	0 - 12–21 17:10 12–22 01:30	0- 12-22 09:50 12-21	17:10 12–22 01:30 12–22 09:50

3.2.8.3. 详情

容器详情页面展示了容器的 yaml 文件:



概览	洋情
args:	
logt	Jerr
seci	listen-address=[\$(IP)]:9100
tls-0	er-
suites=	_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS
ups	m=http://127.0.0.1:9100/
env:	
- name:	
valueF	a
fieldF	
api∖	ion: v1
field	h: status.podIP
image:	I.11.205/k8s-deploy/kube-rbac-proxy:v0.6.0
imageP	olicy: IfNotPresent
name: k	+-rbac-proxy
ports:	
- contai	Port: 9100
hostPo	9100
name:	20
protoc	rcp
resourc	
limits:	
cpu:	
mem	40Mi
reques	
cpu:	
mem	20Mi
termina	MessagePath: /dev/termination-log
termina	MessagePolicy: File

进入单个容器层面,可查看此容器的 CPU、内存、磁盘使用及重启次数等详细信息。

3.2.9. 管理组件监控

平台将 kube-system 命名空间相关的 pod 定义为管理组件,偶发的管理组件异常也将影响整个 k8s 集群的正常运作,所有对管理逐渐的监控也是有很大的必要性。

通过查看管理组件的 pod 状况,判断系统管理组件是否异常。

3.2.9.1. 管理组件列表



概览	详情	告警(159)	节点(9)	命名空间(1	7)	服务(45)	工作负载(58)	Pod(100)) 管理组件
									输入关键词搜索
名称				为	大态 ② 🍷		节点名称		节点IP
calico-kub	e-controllers-	59877c7fb4-wtk57		•	正常		10.10.101.68-master		10.10.101.68
calico-nod	e-6j7s6				正常		10.10.102.93-salve		10.10.102.93
calico-nod	e-dl5rw				正常		10.10.101.72-database-3		10.10.101.72
calico-nod	e-gqkw4			•	正常		10.10.102.91-salve		10.10.102.91
calico-nod	e-jh7qx			•	正常		10.10.101.73-prometheus-d	ata	10.10.101.73
calico-nod	e-k8jnr			•	正常		10.10.101.71-database-2		10.10.101.71
calico-nod	e-IIImx			•	正常		10.10.101.68-master		10.10.101.68
calico-nod	e-pzmzh				正常		10.10.102.92-salve		10.10.102.92
calico-nod	e-tm24z			•	正常		10.10.101.69-gateway		10.10.101.69

3.3. 业务监控

实现业务监控,从客户端调用服务共享平台的服务网关、容器平台内部算法、底层数据库及中间件等。需要支持对上下层平台间服务调用关系及健康度、调用频率等进行全链路监控。

针对于链路调用情况,要求监控数据的分析要快,分析的维度尽可能多。跟踪系统能提供 足够快的信息反馈,就可以对生产环境下的异常状况做出快速反应。分析全面,避免二次开发, 并能提供动态构建业务调用链路监控,以便轻松定位失败点和瓶颈。

3.3.1. 业务列表

通过对业务设定了考核标准,根据不同的健康度指标依照算法考核标准得出业务的考核分数,判定算法优劣。

3.3.1.1. 业务列表



目前针对算法的响应时间、错误率、Apdex 指数和 Pod 运行情况综合评估后,得出算法的运行情况是否满足需求点。如下图:

业务	业务节点											3
新建业务												
业务名称	健康评分 💲	业务状态 👙	所属租户	总请求量 💠	平均响应时间 💲	错误率 💠	Apdex指数	所属集群	pod重启次数	健康检查URL错误数	操作	
69业务	69	正常:5 异常:0 故障:2	69租户	49031	22.27ms	0.06%	1	k8-68-dev	9	0	编辑	删除
System	73	正常:5 异常:1 故障:3	测试租户1	75957	45.41ms	1.22%	1	k8-59-dev,k8- 68-dev	34	0	编辑	删除
OCR	(76)	正常:0 <mark>异文:1</mark> 故障:1	ywx的租户	155960	805.25ms	99.87%	0.5	k8-68-dev	1	0	编辑	删除
测试业务	(81)	正常: 1 异常: 1 故障: 1	ywx的租户	155960	805.25ms	99.87%	0.5	k8-68-dev	1	0	编辑	删除
云上监控	86	正常:7 异常:0 故障:3	MONITORING	613	0.54ms	0%	1	k8-68-dev,k8- 59-dev	23	0	编辑	删除
Default	(86)	正常:1 异常:0 故障:2	测试租户1	0	Oms	0%	0	k8-68-dev	0	0	编辑	删除
bookdemo	90	正常:1 异常:3 故障:0	测试1209	2890	1.5ms	0%	1	k8-159-rel	1	0	编辑	删除
服务测试	92	正常:6 <mark>异常:0</mark> 故障:2	测试租户1	907	1.27ms	0%	1	k8-68-dev	6	0	编辑	删除
hcmine	95	正常:7 异常:0 故障:2	测试租户1	459	2.53ms	0%	1	k8-68-dev,k8- 159-rel	1	0	编辑	删除

在业务列表视图,显示当前项目下的应用列表和应用查询框,每行代表一个业务,此业务 聚合了业务下所有应用。每行显示业务的一些基本信息:

- 1. 业务名称: 配置好的业务名称;
- 健康评分:业务入口的健康度,应用评分,根据响应时间,网络错误率,apdex值指标进行计算;
- 业务状态:业务下业务节点的状态,包含正常、异常、故障三种状态,呈现状态下的 个数值;正常:无任何故障和异常;异常:代码异常、状态码 4xx;故障:状态码 5XX、 建联失败、宕机;
- 4. 所属租户: 业务所属的租户;
- 5. 总请求量: 以业务和时间为维度, 计算业务的被请求的数量;
- 6. 平均响应时间: 所选时间范围内, 业务入口的平均响应时间;
- 7. 错误率:业务入口所有实例发生状态码错误的比率平均值;

第 60 页



- 8. 所属集群:业务里面所涉及到的集群,支持多集群;
- 9. Apdex 指数: 根据业务入口下所有实例的平均响应时间来计算;
- 10. Pod 重启次数:按照业务维度统计业务关联应用的 Pod 重启次数信息
- 11. 健康检查 URL 错误次数:实时统计该业务下所有应用的健康检查错误次数;
- 12. 编辑、删除按钮; 对业务进行编辑删除操作, 删除业务不可恢复;

3.3.1.2. 新建业务

点击新建业务按钮可跳转到新建业务拓扑的页面:



当前视图通过7天的网络调用数据,搭建应用间调用关系,生成拓扑, k8s环境节点是以 工作负责为最小单元,虚拟机环境以 ip:port 为最小单元。根据当前所选的租户,提供给 用户在集群的租户调用关系拓扑并在此拓扑上创建所需业务的能力。可对点击节点名称右 边的图标对节点进行名称修改。

双击所需要加入业务的节点,即可将所选节点加入到业务预创建视图:

第 61 页



in the



选择之后点击保存可弹出下一个页面:

* 业务名称:	OCR		
*租户:	ywx的租户	×]	
* 业务入口:	接口网关 ×		
* 评分规则:	默认规则	×]	



选择业务入口应用,计算业务的健康情况、访问量和健康检查错误数等反应业务健康状态的数据。根据所配的业务,默认选中无上游节点的业务节点作为业务入口,若调整业务入口,则调整后的业务节点为业务入口。

选择评分规则,系统将根据所选评分规则内设置的权重进行健康分数计算。

点击确定之后即可新建一个新的业务。

3.3.2. 业务节点列表

业务节点列表对所有业务的节点进行统一展示,使节点统计更加清晰,如下图:

新建									输入关键词搜索(
名称	所属命名空间	所鳳集群	所属租户	所在业务数 👙	节点类型 🐨	响应时间 🖕	请求量 ≑	错误率 💠	操作
process-exporter	monitoring	k8-68-dev	69租户	2	工作负载	2.49s	72	0%	编辑删除
process-exporter	monitoring	k8-68-dev	MONITORING	2	工作负载	2.49s	72	0%	编辑 删除
testdemo1	testapp	k8-68-dev	ywx的租户	2	工作负载	804.7ms	14865	99.84%	编辑 删除
testdemo2	testapp	k8-68-dev	ywx的租户	2	工作负载	599.75ms	18264	0%	编辑删除
calico-node	kube-system	k8-68-dev	测试租户1	1	工作负载	482.06ms	486	0%	编辑删除
process-exporter	monitoring	k8-159-rel	测试1209	2	工作负载	299.9ms	40	0%	编辑 删除
process-exporter	monitoring	k8-159-rel	测试租户1	2	工作负载	299.9ms	40	0%	编辑 删除
node-exporter	monitoring	k8-68-dev	69租户	2	工作负载	265.1ms	306	0%	编辑删除
node-exporter	monitoring	k8-68-dev	MONITORING	2	工作负载	265.1ms	306	0%	编辑 删除
node-exporter	monitoring	k8-159-rel	测试1209	2	工作负载	44.28ms	160	0%	编辑 删除

列表提供了对应用节点的编辑(k8s节点不可删除)、删除功能。

点击新建按钮, 跳转至新建应用界面:



* 名称:		
* 类型:	微网关	\sim
* VIP:	请输入IP或网段	
调用外部应用:		

新增应用类型提供了四种应用类型:F5设备、SLB、微网关、虚拟机应用。

F5设备、SLB、微网关三类需输入对应VIP,单平台匹配到对应的VIP地址将将 ip 替换为对应的名称,从而用具体的设备名称替换抽象的 ip:port。

虚拟机应用需输入对应的关联端点(ip:port),虚拟机应用将在1.3.3.6节归并展开详细说明:

新增应用		×	
* 名称:			
* 类型:	虚拟机应用	~	
	F5 设备		
* 关联端点:	SLB		
	微网关		
	虚拟机应用		
调用外部应用:			

3.3.3. 业务详情

在业务列表也选择一个业务名称点击进入即可跳转到业务详情页面,业务详情页面划分为:调用拓扑、概览、服务、工作负载、告警5个模块。

3.3.3.1. 调用拓扑

此模块将显示此业务的节点调用拓扑图:

	接口网关	• OCR	*****	
& k8-68-de				ocr &k8-68-de
故障	响应时间 304.3ms	错误率 0%	请求量 15.94万次	异常

第65页



如图所示:

拓扑图上可展示业务中客户端节点的名称、所属集群、状态、节点之间的网络调用、响应时间、错误率、请求量等数据。

点击当个节点,可显示此节点做为服务端所关联的网络数据:



包括: 响应时间、错误率、请求量健康检查错误数。

点击回溯按钮,可回溯所选时间内的业务节点数据拓扑,支持自动播放:







3.3.3.2. 概览

概览模块统一将统计该业务的 CPU、内存、磁盘、健康检测 URL 错误数。

调用拓扑 ⑦ 機宽 服务(0) 工作负载(0) 告輩(0)	应用(0)	
CPU使用量内存使用量	磁盘使用量	健康检查URL错误致
监控数据 ∨		
CPU使用量	内存使用量	磁盘使用量
智无数据	智无数据	智无数据
入口应用请求量	入口应用错误率	入口应用响应时间
5,000	100%	15
	80%	800ms
	40%	400ms
1,000	20%	200ms -
0	0% 12-31 09:44 12-31 09:54 12-31 10:04 12-31 10:14 12-31 10:24 12-31 10:34	0 12-31 09:44 12-31 09:54 12-31 10:04 12-31 10:14 12-31 10:24 12-31 10:34

并根据响应时间、错误率、吞吐量三个纬度筛选出响应的 TOP5 应用。

第 67 页





3.3.3.3. 服务

点击服务 Tab 可跳转入服务列表页面,此页面所展示的是此业务内涉及到的服务。

名称	命名空间	所属集群	所在业务数	服务类型	晌应时间 🝦	请求量 👙	错误率 💠
testdemo1	testapp	k8-68-dev	2	k8s服务	803.04ms	163852	99.85%
testdemo2	testapp	k8-68-dev	2	k8s服务	600.05ms	172736	0

如图所示:

服务列表包括:服务名称、命名空间、所属集群、所在业务数、服务类型、响应时间、请 求量、错误率等指标。

3.3.3.4. 工作负载

点击工作负载 Tab 可跳转入工作负载列表页面,此页面所展示的是此业务内涉及到的工作 负载。

名称	命名空间	所属集群	所在业务数	工作负载类型	响应时间 🝦	请求量 ⇔	错误率 💠
testdemo1	testapp	k8-68-dev	2	deployment	803.06ms	161981	99.84%
testdemo2	testapp	k8-68-dev	2	deployment	600.04ms	170963	0



如图所示:

工作负载列表包括:工作负责名称、命名空间、所属集群、所在业务数、工作负载类型、 响应时间、请求量、错误率等指标。

3.3.3.5. 告警

点击告警 Tab 可跳转入告警列表页面,此页面所展示的是此业务内涉及到的告警。

告警内容	告警对象	告警类型 🛛 🔻	告警等级 🌲	所属业务	通知对象	告警时间 💠	状态 🛒	操作
k8-68-dev 集群下te	testdemo1- 7445745558-thswc	Pod告警	P1	测试业务 OCR	RDIO	2021-12-28 17:00:00	◎ 未处理	处理
k8-68-dev 集群下te	testdemo1- 7445745558-thswc	Pod告警	P1	测试业务 OCR	RDIO	2021-12-28 16:54:00	◎ 未处理	处理
k8-68-dev 集群下te	testdemo1- 7445745558-thswc	Pod告警	P1	测试业务 OCR	RDIO	2021-12-28 16:44:00	● 未处理	处理
k8-68-dev 集群下te	testdemo1- 7445745558-thswc	Pod告警	P1	测试业务OCR	RDIO	2021-12-28 16:29:00	• 未处理	处理
k8-68-dev 集群下te	testdemo1- 7445745558-thswc	Pod告警	P1	测试业务OCR	RDIO	2021-12-28 16:23:00	◎ 未处理	处理
k8-68-dev 集群下te	testdemo1-	Pod告警	P1	测试业务 OCR	RDIO	2021-12-28	• 未处理	处理

如图所示:

告警列表包括:告警内容、告警对象、告警类型、告警等级、所属业务、通知对象、告警 时间、状态等指标。

3.3.4. 节点详情

在调用拓扑上点击当个节点的弹出框中,再点击查看更多可跳转如当节点的详情界面:





3.3.4.1. 概览

进入单个节点(工作负载)层面,可查看节点 CPU、内存、磁盘等资源使用情况及请求量、 错误率、相应时间等网络数据。

查看节点下 pod 发起与接收网络请求及 pod 异常数据,包括: pod 发起的错误请求, 相应时间超过阈值的慢请求、连接失败、收到的错误请求、收到的慢请求、包追踪、健康 检查、pod 消亡原因及消亡时间8个层面。



云监控(HC-CloudMonitor)产品说明书

CPU使用量 0.08核		内存使用量 41.94M	磁盘使用量 255.54M	健康检查URL错误数 33551	
监控数据 > CPU使用量		内存使用量		磁盘使用量	
0.15枝 0.12核 0.09核 0.06核 0.03板 0板 12-22 22:0	m.	47,68M 38,15M 28,61M 19,07M 9,54M 9,	UP-20 0223 12-20 04.33 12-20 0643 12-20 0853	2.330 1.860 1.46 953.67M 476.84M 0 12-22 2203 12-23 00-13 12-23 02.23 12	-23 04.33 12-23 06.43 12-23 08.53
请求量		错误率		响应时间	
18,000 15,000 9,000 6,000 3,000 0 12-22 22:0	0 12-2210030 12-2210220 12-2210420 12-221060	100% 0 80% 80% 40% 20% 0% 12-22 22:00 12-23 00:10	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	1,000ms 800ms 400ms 200t-12-23 06 9 minE131 Mi 200ms 12-22 00 12-23 00:10 12-23 02:00 12	-22 04.30 12-23 06.40 12-23 08.50
工作负载下P	od的发起和接收情况				URL过滤 导出 输入关键词搜索 Q
○ 发起错误 URL	请求 E 发起微请求 ● 连接失败 ● © 收到错 <3s ≑	美请求● 🖾 收到營请求● 🖾 包追踪 3s - 5s 💠	> 5s 👙	总量 🗘	
/test	2050458	0	0	2050458	
					第 1-1 条/总共 1条 < 1 >
异常事件 健康检查	POD消亡				健康检查 输入关键词搜索 Q.
名称	4xx次数	5xx次数	រ គ្រ	总次数 💿 🗘	
/test	0	2048952	204	3952	
					第 1-1 条/总共 1条 (1) >

1) 发起的错误请求:

监控数据 >			
工作负载下Pod的发起和接收情况			URL过滤 导出
○ 发起错误请求 ⑧ ▷ 发起慢请求 ⑧ ▲ 连接失败 ④ ○ 收到错误请求 ⑧ □ 收到慢请求 99+ □ 包追踪			输入关键词搜索 Q
URL	4xx次数 ≑	5xx次数 👙	总次数 🖕
/apm/app/slowMethod/methodView	1	0	1
/apm/app/slowMethod/top5SlowMethod	1	0	1
/trans/Log/appLogList	1	0	1

统计了 4xx 错误, 5xx 错误及总错误次数。

点击某个错误请求的 url,可下钻查看单个 url 报文分析层面,如下图:



错误请求列表	请求调用链 http		返回码: 401 大小: 2548 耗时: 8.84ms
请输入关键字搜索 Q	項論		
時向时间 ・ 明点大小 / 友生时间 + 特选 ¹ http://192.168.84.120-9900/apm/app/slowMethod/me target=yanshi 源語,nginx-ingress-controller-h影響	Aban 192.188.106.128.42310 服务名院: nginx-ingress-controller-hs POD名限: nginx-ingress-controller-9649/46d-kmt 76	xiisti)	日分類 http://L88.84.120-9900/apm/ap/alowMethod/methodView?target=yanshi_yans hi_ga&agentid=&type=&&from=2 192.168.84.120-9900
□ 401 dd 254B ⊙ 8.84ms 2022-01-04 10:20:30	请求 响应 时间线 包追踪 ③		
	阶段 时间线	响应时间	数据大小
	造报0		
	TCP握手 I SSL握手	128.27µs Ons	
	请求仰应		
	发送请求	360.21µs	671B
	等待响应 内容下载	8.34ms 8.13µs	254B
	总计	8.84ms	9258

根据错误请求列表可判断此 url 发生错误请求的相关信息。

根据请求调用链可判断此 url 发生错误请求时的调用方与被调用方的信息,由此类信息可 判断网络请求报文级别的异常原因。包括请求、响应、时间线及包追踪。时间线展示为此次连 接各个阶段的耗时。

2) 发起的慢请求:

工作负载下Pod的发起和接收情况				URL过滤 导出
⊙发起错误请求 3			3	俞入关键词搜索 Q
URL	< 3s 💠	3s - 5s 🖕	> 5s 💠	总量 ≑
/prometheus	851	0	140	991
/apmServer-sl/overview/unprocessed-alarm/list	7	0	0	7
/api/v1/query	1	0	1	2
/api/v1/query_range	2	0	0	2

第 1-4 条/总共 4条 💦 📘 🔊

统计了发生慢请求时 < 3s, 3-5s, > 5s 及慢请求总数。

点击某个错误慢的 url,可下钻查看单个 url 报文分析层面,如下图:


慢请求列表	请求调用链 http		返回码: 200 大小: 3.38K 耗时: 955.08ms
请输入关键字搜索 Q	海道	E MUNH	•
响应时间◆ 响应大小◆ 发生时间◆ 筛选 3	192.168.38.132:33530	发送成功 http://192.168.38.158:9201/test	?redis=1&sleep=600&respbyte=3300
http://192.168.38.158:9201/test? redis=1&sleep=600&respbyte=3300	股务名称: testdemo1 POD名称: testdemo1-57fc8b64bd-cpg2m	192.168.38.158:9201 服务名称: testdemo2	
避瑞:testdemo1服务 □ 200 inl 3.38K ◎ 955.08ms 2021-12-22 22:13:54			
http://192.168.38.158:9201/test? redis=T&sidep=500&respbyte=3300 夏蓮:testdemo1服务 200	(1)		
http://192.168.38.137:9201/test? redis=1&sleep=500&respbyte=3300 遵题:testdemo1陽务 □ 200 네 3.38K © 914.88ms 2021-12-23 09:50:26	请求 < HTTP/1.1 200 OK Content-Type: text/plain; charset=utf-8 Date: Wed, 22 Dec 2021		
http://192.168.38.137:9201/test? redis=1&sideep=600&respbyte=3300 @st.testdemo1@sf 200 =i 3.38K © 904.28ms 2021-12-23.08:54:56			
http://192.168.38.158:9201/test? redis=1&sleep=600&respbyte=3300			

根据慢请求列表可判断此 url 发生慢请求的相关信息。

根据请求调用链可判断此 url 发生时的调用方与被调用方的信息,由此类信息可判断网络请求报文级别的异常原因。包括请求、响应、时间线及包追踪。

3) 连接失败:

监控数据 >							- 1
工作负载下Pod的发起和接收情况						URL过速	导出
⊙ 发起错误请求 3 □ 发起慢请求 4	▲ 连接失败4 ⊙ 收到银	错误请求 <mark>32</mark> □ 收到性	豊请求 99+ 🛛 🗊 包追踪			输入关键词搜索	٩
源POD	源IP		目标IP	目标POD	目标服务	连接数	
nginx-ingress-controller-9649f46d-km.	192.168.106.128		192.168.240.74	prometheusbeat-deployment-7f8644d		6	
nginx-ingress-controller-9649f46d-km.	127.0.0.1	•	0.0.0.0	0.0.0.0:443		1	
nginx-ingress-controller-9649f46d-km.	127.0.0.1		0.0.0.0	0.0.0.0:80		1	
nginx-ingress-controller-9649f46d-km.	127.0.0.1		0.0.0.0	0.0.0.0:8181		1	

统计了连接失败时源 pod、源 IP、目标 IP、目标 POD、目标服务及连接数。

点击连接失败的单个 pod, 可进入 pod 发生网络请求发生连接失败的具体信息:



连接失败列表		连接信息		
请输入关键字搜索	٩	源锑		· · · · · · · · · · · · · · · · · · ·
发生时间♥	筛选习	192.168.38.156:38032	连接失败 目的端 10.98.124.4	9201
遼端: testdemo1 目的端: testdemo2 源IP: 192.168.38.156 目标IP: 10.98.124.4 请求时间: 2021-12-23 10:18:39	,	ND 完 年版: testdemo1 POD名称: testdemo1-57fc8b64bd-cpg2m 客方间线 包追踪 ②	副务名称:1	stdemo2
源靖: testdemo1 目的靖: testdemo2 源(P: 192.168.38.156 目杨(P: 10.98.124.4 译: 地球时候 0.021 10.02 10:18:20		R介段 8月间8条 155日80	响应时间 数据大小	
讀來時间:2021-12-23 10:18:39 源端: testdemo1 目的端: testdemo2 源IP: 192.168.38.156		TCP握手 I SSL握手	163.62µs Ons	
目标IP: 10.98.124.4 请求时间: 2021-12-23 10:18:39 源靖: testdemo1 目的靖: testdemo2 源IP: 192.168.38.156 目标IP: 10.98.124.4		请求,响应 发送请求 等待响应 内容下载	Ons O Ons O	
请求时间:2021-12-23 10:18:39 源語: testdemo1 目的源: testdemo2 源IP: 192.168.38.156 目频IP: 10.98.124.4 请求时间: 2021-12-23 10:18:39		A&H	327.23µs 0	

如图所示:

可根据连接失败列表记录连接失败历史的信息。可根据连接信息判断连接的源段与客户端的信息包括连接发生时所统计的连接、请求、响应等时间线来判断此连接异常的原因及异常定位。

4) 收到错误请求:

监控数据 > 工作负载下Pod的发起和接收情况 ◎ 发起错误请求 ⑧ ◎ 发起偿请求 ● 连接失败 ◎ 收到错误请求 ③ ◎ 收到偿请法 ●● □ 包追踪			URL过滤 号出 输入关键词搜索 Q
URL	4xx次数 👙	5xx次数 👙	总次数 💠
/no_exist_index/_search	1424	0	1424
/apm2.0-yanshi_yanshi_ga-javamanagementdata . ¥/_search	0	71	71
/apmServer-sl/scc-cluster/list/k8s	0	5	5
/apmServer-sl/business-info/list	0	4	4
/apmServer-sl/overview/unprocessed-alarm/list	0	4	4
/apmServer-sl/overview/recent/pod-event	0	4	4
/aomServer-si/alarm/list	ō	3	3

统计了 4xx 错误, 5xx 错误及总错误次数。

点击某个错误请求的 url,可下钻查看单个 url 报文分析层面,如下图:



Apple bit is apple b				
	错误请求列表	请求调用链 http		返回码: 503 大小: 265B 耗时: 1.73ms
	请输入关键字搜索 Q			目的端
http://10.10.16.95.39200/pam2.0- yansh_ua-javannagementdata-1	响应时间 ◆ 响应大小 ◆ 发生时间 ◆ 筛选 ②	源端	发送成功	http://10.101.69:39200/apm2.0-yanshi_yanshi_ga-javamanagementdata-1/_sea rch?typed_kevs=true&ignor
加速の12000 1.2mm 2022-01-04 07:18-04 http://10.101.69:39200/pem2-0- yanshi_yansh_ga-javannangementdat-1 前後	http://10.10.101.69:39200/apm2.0- yanshi_yanshi_qa-javamanagementdata-1	192.106.106.133.32.104 服务名称: testser POD名称: testy-pod	(e)	10.10.101.69-39200 服务名称: nginx-ingress-controller-hs
Imput/10.10.10.69.39200/pm2.0- yanbl_vanshl_a-javamanagementdata1 Imput Imput <th< td=""><td>激躁:testser服务 © 503 由 265B © 1.73ms 2022-01-04 07:18:45</td><td></td><td></td><td></td></th<>	激躁:testser服务 © 503 由 265B © 1.73ms 2022-01-04 07:18:45			
yanhi yanhi yanshi ga-javamangemetda 1 phg pimid	http://10.10.101.69:39200/apm2.0-	请求 响应 时间线 包追踪 ⑦		
http://10.10.16.69.39200/pm2.0- yanshi_uanshi_ua-javananagementdata-1 ž#0 J#testwer#JM CP#F Ons 0 30 / 2558 © 2.26ms 2020-01-04 07.1724 Ons http://10.10.16.9-39200/pm2.0- yanshi_uanshi_ua-javananagementdata-1 J#xeminumenter filter	yanshi_yanshi_qa-javamanagementdata-1 © 503 al 265B © 4.22ms 2022-01-04 07:18:44	阶段 时间线	响应时间	数据大小
万公単小 丁CP導手 Cn 1000 000 000 1000 000 000 1000 000 000 http://10.10.10.696.392.00/pam2.d- yanshLyansh_qa-javamangementdata1 55.4億 0ns <u>jaketus day</u> <u>jaketus day</u>	http://10.10.101.69:39200/apm2.0- yanshi_yanshi_qa-javamanagementdata-1	连接0		
SSL #F SSL #F Ons http://10.10.16.96/39200/pmm2.4- yanshi_yanshi_qa-javamanagementdata-1 ist/min ist/min ###testker##F ist/min ist/min 1.02K ##testker##F SSL #F 9.54µs 1.02K #ttp://10.10.169/39200/pmm2.4- yanshi_ga-javamanagementdata-1 Käitik 9.54µs 1.02K #ttp://10.10.169/39200/pmm2.4- yanshi_ga-javamanagementdata-1 Käitik Introversite 1.02K	源端:testser服务	TCP握手	Ons	
http://10.10.1689.39200/apm2.0- yanshi_yanshi_ga-javamanagementdata-1 请求周回 選進testser服务 1050 2022-01-04 07.17:42 友送请求 9.54µs 1.02K 等待助应 1.67ms http://10.10.10.689.39200/apm2.0- yanshi_ga-javamanagementdata-1 内容下载 51.38µs 2658	© 503 and 2658 © 2.26ms 2022-01-04 07:17:42	SSL握手	Ons	
ygansh_ga-javamanagementdata-1 šityingd Jättetsteräßi §2,64µ5 1.02K S03 al 2658 © 1.96ms 2022-01-04 07.17/42 Žättetsteräßi 1.67ms http://10.10.10.69:39200/apm0.2	http://10.10.101.69:39200/apm2.0-			
選進(testke)協務 2022-01-04 07.17:42 次送消水 1 0.54μs 1.02K 日 50 a (2658) © 1.95ms 2022-01-04 07.17:42 等待時应 1.67ms http://10.10.10.69:39200/apm2.0- yanshi_qa-javamanagement/data-1 内容下载 51.38μs 2658	yanshi_yanshi_qa-javamanagementdata-1	请求向应		
103 al 2058 © 139mm 2022-01-04 0/17/42 16.7ms 16.7ms 16.7ms http://10.10.169:39200/apm2.0- yanshi_ua-javannagementdata-1 内容下载 61.38µs 2658	源端:testser服务	发送请求	9.54µs	1.02K
http://10.10.101.69:39200/apm2.0- 内容下载 51.38µs 2658	© 503 al 2658 © 1.95ms 2022-01-04 07:17:42	等待响应	1.67ms	
yanshi_yanshi_qa-javamanagementdata-1	http://10.10.101.69:39200/apm2.0-	+00T#2	51.20	0000
通偿tacted 和条	yanshi_yanshi_qa-javamanagementdata-1	內谷下戰	51.36µS	2008
10 xm, rr.2010c1 x0/34	源端:testser服务			
□ 503 🖬 2858 © 1.84ms 2022-01-04 07:17:42 总计 1.73ms 1.28K	□ 503 ad 265B ○ 1.64ms 2022-01-04 07:17:42	总计	1.73ms	1.28K
nttp///to.uo.tu.os/szcuughttcu- vanbi vanbi, naki, nakiawananomentda-1	nup://10.10.10.101.09-39200/apm2.0-			
DT03 #1259 423m 2022-01-04 07:17/41	□ 503 al 265B ○ 4.23ms 2022-01-04 07:17:41			

根据错误请求列表可判断此 url 响应错误请求的相关信息。

根据请求调用链可判断此 url 发生错误请求时的调用方与被调用方的信息,由此类信息可 判断网络请求报文级别的异常原因。包括请求、响应、时间线及包追踪。时间线展示为此 次连接各个阶段的耗时。

5) 收到慢请求:

监控数据 >				Ś
工作负载下Pod的发起和接收情况				URL过滤 导出
 ○ 发起错误请求 ◎ 发起偿请求 ◎ 收到错误请求 ◎ 收到错误请求 ◎ 收到错误请求 ◎ 收到错误请求 ◎ 收到错误请求 			1	喻入关键词搜索
URL N	< 3s 🍦	3s - 5s 💠	> 5s 👙	总量 ≑
/_cat/indices	1924	452	78	2454
/prometheus	1005	86	140	1231
/agent/status/heart	253	42	201	496
/tcp-dump/getShellScript	190	72	48	310
/hcmine/config/update	130	36	67	233
/apmServer-sl/business-info/list	127	17	15	159
/apmServer-sl/business-info/status	72	6	6	84

统计了发生慢请求时 < 3s, 3-5s, > 5s 及慢请求总数。

点击某个错误慢的 url,可下钻查看单个 url 报文分析层面,如下图:



慢请求列表	请求调用链 http		返回码: 200 大小: 3.97K 耗时: 8.33s
请输入关键字搜索 9			日的海
响应时间→ 响应大小+ 发生时间+ 筛选 3	源端	发送成功	http://10.10.101.69:39200/_cat/indices?h=Index&index=*
http://10.10.101.69:39200/_cat/indices? h=index&index=* 200 _d 87K @ 8.33s2022-01-04.08:32:38	POD24%: testy-deployment-85ddcf6f58-pjizm	4	2010.10.101.69:39200 服务名称: nginx-ingress-controller-hs
http://10.10.101.69:39200/_cat/indices? h=index&index=*	请求 响应 时间线 包追踪 ⑦		
源端:testser服务 □ 200	阶段 时间线	响应时间	数据大小
http://10.10.101.69:39200/_cat/indices? h=index&index=apm2.0+*-** 那葉testser服务 □ 200 ==3.91K © 7.73s 2022-01-04 08:32:38	连续0 TCP獨手 SSL欄手	Ons Ons	
http://10.10.101.69:39200/_cat/indices?	请求/响应		
© 200 all 3.91K © 7.58s 2022-01-04 01:12:38	发送请求	3.93µs	174B
http://10.10.101.69:39200/_cat/indices?	等待响应	8.33s	
h=index&index=apm2.0-*-*-*	内容下载	106.78µs	3.97К
200	总计	8.33s	4.14K

根据慢请求列表可判断此 url 收到慢请求的相关信息。

根据请求调用链可判断此 Url 发生时的调用方与被调用方的信息,由此类信息可判断网络请求报文级别的异常原因。包括请求、响应、时间线及包追踪。

6) 包追踪

100 127	四田姓 ○ 牛酸(4)	公司任礼						
1946,345		110022011						
工作负	载下Pod的发起和接收情况							URL过滤 导出
⊙ 发起	显错误请求 💷 发起慢请求 🚺 🖮 连接约	▶ 1 ● 收到错误请求 2 ■ 收	到慢请求 2 🗈 包追踪					
	源IP:		源Pod:			发起时间:	2021-12-28 14:10:3 ~ 3	2022-01-04 14:10:3!
	目标IP:		目标Pod:					重置 查询
	skb_addr	源POD	源IP	目标POD	目标IP	目标服务	通过网卡数量	发起时间
+	18446612147123890936	prometheus-k8s-0	192.168.38.191		10.10.101.73		1	2022-01-04 14:09:37
+	18446612149401579520	prometheus-k8s-0	192.168.38.191		10.10.101.73		1	2022-01-04 14:09:22
+	18446612149091721216	prometheus-k8s-0	192.168.38.191	10.10.101.73:730-kubelet	10.10.101.73		1	2022-01-04 14:09:22
+	18446612144828801272	prometheus-k8s-0	192.168.38.191	10.10.101.73:730-kubelet	10.10.101.73		1	2022-01-04 14:09:20
+	18446612144828808952	prometheus-k8s-0	192.168.38.191		10.10.101.73		1	2022-01-04 14:09:09
+	18446612149091721216	prometheus-k8s-0	192.168.38.191		10.10.101.73		1	2022-01-04 14:09:07

需在1.3.10.3 探针管理中探针开启包追踪,开启包追踪之后,可抓取网卡的信息及通过网 卡的数据包信息。包追踪功能开启时对性能需求较大,勿长时间开启。

可展示 skb_addr、源 pod、源 IP、目标 pod、目标 IP、目标服务、通过网卡数量、发起时间等信息。可通过各个信息关键字进行筛选查询。

点击单条数据包可展示数据包相关网卡的信息:



作负载	t下Pod的发起和接收	情况							URL过滤 号 E
n0 (2.168.)	38.191		》收到错误请求2 🛛	收到慢请求 2 🛛 包追踪					
器网卡 线数据包	3			源Pod :			发起时间:	2021-12-28 14:19:5 ~ :	2022-01-04 14:19:5: 📋
	源	目标		目标Pod:					重置 重询
MAC	16:7b:bf:43:f6:42	ee:ee:ee:ee:ee	POD	源IP	目标POD	目标IP	目标服务	通过网卡数量	发起时间
Р	192.168.38.191	10.10.101.73	ometheus-k8s-0	192.168.38.191		10.10.101.73		1	2022-01-04 14:18:39
端口	50418	9256							主机2
	eth0 prometheus-ki	3s-0							↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔ ↔
+	1844661214471828	6080	prometheus-k8s-0	192.168.38.191		10.10.101.73		1	2022-01-04 14:18:37
	1844661214471828	3776	prometheus-k8s-0	192 168 38 191	10.10.101.73:730-kubelet	10 10 101 73		1	2022-01-04 14:18:22

如图,悬浮某个网卡图表上,可弹出此网卡的信息及出栈数据包。

7) 健康检查

异	常事件					健康检查
1	建康检查② POD消亡 11				输入关键词搜索	Q
	名称	4xx次数	5xx次数	调用总次数 ⑦ 🝦		
	/api/v1/query	0	1	26610		

展示健康检查相关 url、4xx、5xx 及调用总次数。

8) Pod 消亡

异常事件 健康检查 POD射亡●			输入关键词搜索	健康检查 Q
名称	消亡原因	发生时间		
testdemo2-5b7b5d88ff-5rt95	正常退出(null)	2021-12-29 11:17:19		

展示相关 pod 消亡的原因及消亡时间。

3.3.4.2. 调用链

进入调用链层面, k8s 可查看 pod 调用 pod(含 service 调用)的调用状态及调用数据,

可判断异常调用发生时 pod 间的调用关系及调用双方的状态,虚拟机可查看 ip:port 调用

ip:port 的调用关系。以下介绍 k8s 环境:





点击线可展示调用双方的信息及客户端的网络数据包括:响应时间、错误率、请求量。 点击某个节点可显示作为服务端的网络数据:包括响应时间、错误率、请求量。





3.3.4.3. 分层拓扑

关于分层拓扑,可见分层拓扑,此处不做详述。

3.3.5. 租户拓扑

集群管理往往是以租户纬度进行隔离的。站在租户角度对整个权限范围内的应用访问 情况进行把控是非常有意义的。因此平台提出了一个租户拓扑的模块,对租户权限范围内 的应用访问情况进行了图形化展示,租户相关的应用访问情况在拓扑图上一目了然。如下 图为单个租户内的应用访问拓扑:



节点与节点之间的有向线段的方向即表示应用之间的调用方向。

线上的小黑点多少表示客户端到服务端的流量大小,黑点越多流量越大。

每个节点如果有故障或者异常则显示红色故障及黄色异常。

点击节点间的调用线段,弹出框显示客户端到服务端的调用数据:响应时间、错误率请求量。如图:





点击当个节点,弹出框则展示服务端的网络数据:响应时间、错误率、请求量、健康检测错误数,如下图:



3.3.6. 归并

在虚拟机环境中,应用是以 ip:port 的形式对外提供服务的,当同一个功能应用拥有多个 ip:port 时,我们往往会对一类的 ip:port 进行抽象归类,平台将此归类描述为"归并"。当对 某些 ip:port 进行归并后,拓扑图上将做出对应的归并变化,更加简练。归并包括全局归并、 业务内归并两个归并纬度。

3.3.6.1. 全局归并



i

全局归并:归并之后对所有新建业务拓扑、租户拓扑都生效。

在1.3.3.2节业务节点列表新建时有个新建虚拟机应用的功能,此功能即为新建全局归并 应用节点。如图:

* 名称:	虚拟机应用	
* 类型:	虚拟机应用	
	格式为IP:PORT	
* 关联端点:	10.102.157:9100	
	10.102.157:9256	e
	+ 添加端点	
调用外部应用:		

将两个 ip:port 归并为一个名称为:测试虚拟机应用的应用节点。点击保存之后在节点列表中显示:

业务 业务节点	l								
新建									12 O Q
名称	所属命名空间	所属集群	所属租户	所在业务数 💲	节点类型 🔻	响应时间 🝦	请求量 ≑	错误率 💠	操作
虚拟机应用	VM_NS_DEFAULT			0	虚拟机应用	0	0	0%	編辑 删除
								第 1-1 条/总共 1条	1 10多/页 /



租户拓扑显示:



新建业务拓扑:



点击蓝色框可展开收缩,默认为收缩。

3.3.6.2. 业务归并

业务归并:归并后只对本业务拓扑生效,其他业务、租户拓扑不影响。对比全局归并,生效范围只限制与本业务。



归并流程:

1. 双击选中要归并的节点。

2. 点击右上角归并按钮,弹出左上角归并菜单。

3. 输入归并的名称,类型默认为虚拟机应用,双击拓扑以选中的节点,图中黄色节点,

即可自动添加到左侧关联端点 ip:port 输入框内。如下图:



点击保存即可保存设置:





如图:虚机业务归并节点为业务归并之后的节点,虚拟机应用为全局归并之后的节点。展 开可展示归并内部具体的端点:



点击取消归并可取消业务节点归并。

3.4.应用监控

应用组件服务架构的复杂度越来越高,且微服务按照不同的维度进行拆分部署,一次请求 往往需要涉及到多个服务。应用组件构建在不同的软件模块集上,这些软件模块,有可能是由 不同的团队开发、可能使用不同的编程语言来实现、有可能布在了几千台服务器,横跨多个不 同的数据中心。因此,就需要一些可以帮助理解系统行为、用于分析性能问题的工具,以便发 生故障的时候,能够快速定位和解决问题。

3.4.1. 应用列表



应用列表 应	用头例 扣扑	9/151								C 30分钟
根据应用名称搜	索	Q								
应用名称	健康指数 ⑦	应用状态 ②	探针状态 ⑦	应用语言	响应时间 🕜 🝦	Apdex指数 ③ 💠	吞吐量 ≑	错误率 ⑦ 💠	操作	
gateway	(12)	正常:1 异常:0 宕机:0	启动:1 暂停:0 熔断:0 掉线:0	Java	5.16s	0.33	80次	35%	详情	删除
bookdemo	77	正常:1 异常:3 宕机:0	启动:4 暂停:0 熔断:0 掉线:0	Java	4.66s	0.96	567次	8.46%	详情	删除
eurekaclient	(18)	正常:0 异常:1 宕机:0	启动:1 暂停:0 熔断:0 掉线:0	Java	3.95s	0.47	38次	31.58%	详情	删除
eurekaclient	24	正常:0 异常:1 宕机:0	启动:1 暂停:0 熔断:0 掉线:0	Java	3.88s	0.5	68次	26.47%	详情	删除
agent-demo	50	正常:0 异常:1 宕机:0	启动:1 暂停:0 熔断:0 掉线:0	Java	2.29s	0.48	58次	0	详情	删除
application	0	正常:0 异常:0 宕机:2	启动:0 暂停:0 熔断:0 掉线:2	Java	374.9ms	0.95	38次	36.84%	详情	删除
bookdemo-k8s	99	正常:1 异常:1 宕机:0	启动:0 暂停:2 熔断:0 掉线:0	Java	214.86ms	0.93	42次	0	详情	删除
testk8s	100	正常:0 异常:1 宕机:0	启动:1 暂停:0 熔断:0 掉线:0	Java	14.96ms	1	592次	0.17%	详情	删除
gatewayhttps	100	正常:0 异常:1 宕机:0	启动:1 暂停:0 熔断:0 掉线:0	Java	6.92ms	1	26次	0	详情	删除
wuxianju	(100)	正常:1 异常:0 宕机:0	启动:1 暂停:0 熔断:0 掉线:0	Java	0.95ms	1	720次	0	详情	删除

如上,在应用列表视图,显示当前项目下的应用列表和应用查询框,每行代表一个应用, 此应用聚合了应用下所有实例。每行显示应用的一些基本信息:

1. 应用名称:探针获取的应用名称,聚合了应用下所有实例,此处不显示 IP 和端口号。

应用健康指数:非容器应用根据应用内所有实例的健康指数计算平均值;容器应用只根据活着的实例计算健康指数。评分>80分为绿色,60-80分为黄色,<60分为红色。

 应用状态:显示应用状态。显示应用下各状态的实例总个数。应用状态包括:正常、 异常、宕机。

 探针状态:显示应用下各探针状态的实例个数。探针状态包括:启动、暂停、熔断、 掉线。达到用户配置的熔断条件时,探针会熔断。实例宕机时,探针状态为掉线。

5. 应用语言:显示应用使用语言,支持 java、.net、PHP 语言的应用。

6. 响应时间:应用下所有实例响应时间平均值。

7. Apdex 指数: 衡量服务性能的标准, 根据应用下所有实例的平均响应时间来计算。

8. 吞吐量:应用访问次数。

9. 错误率:应用状态码大于等于 400 的请求次数除以总请求次数。

第 85 页



10. 详情/删除按钮:关键的按钮,点击可进入目标应用的详情页,所监控数据都在这里展示;删除应用会删除应用下所有实例,可在设置中恢复。

11. 点击表格行中除删除按钮以外的区域,都可进入该行应用的详情页。

3.4.2. 应用实例

应用列表	应用实例	拓扑视图								⑤ 30分钟	
● 健康指数>=80分 ● 健康指数60-80分 ● 健康指数<60分											
bookdem	o	4	bookdemo-k8s	2	application	2	wuxianju	1	gatewayhttps		
eurekacli	ent-device	1	eurekaclient-user	1	testk8s	1	agent-demo	1	gateway		

如上,在应用实例视图,显示的是当前项目下的聚合实例的应用标签,即将应用下所有实例按照应用名称合并为一个标签。

标签名称后面的数字代表使用该标签的实例数量,鼠标上移时会展示该标签各状态下的实例数量,当健康指数>80分为绿色,60-80分为黄色,<60分为红色。

点击某个标签,会分页展示使用该标签的实例列表和实例查询框,如下图:

应用契例 施用实例 拆扑视图 C 305 应用实例 / bookdemo-k8s										
根据实例名称搜索 Q										
实例名称	健康指数 ②	实例状态 ②	探针状态 ②	应用语言	响应时间 ⑦ 🖕	Apdex指数 ② 👙	吞吐量 💠	错误率 🗇 👙	操作	
bookdemo-56b98bbf58-tdsts@192.1	98	 异常 	 暂停 	Java	248.5ms	0.92	36次	0	详情	编辑 删除
bookdemo-56b98bbf58-fcn6w@192	100	• 正常	 ● 暂停 	Java	13ms	1	6次	0	详情	编辑 删除

实例列表展示参数与应用列表一致:实例名称、健康指数、实例状态、探针状态应用语言、 响应时间等指标。点击左上角的标签列表按钮,可返回;实例列表后面的详情/删除按钮功能 与应用列表的按钮功能一致。注意:实例宕机后,健康指数=0。

3.4.3. 拓扑视图

第86页



在拓扑视图,默认显示当前项目最近 30 分钟内的应用拓扑情况,以一个测试项目为例,拓扑图显示如下:



拓扑图显示了客户每一次点击的链路详情和所选时间段内对每个应用访问的平均响应时间。界面分为左上角状态图标、右上角功能图标、中间拓扑图展示及右侧节点归并展开按钮。

- 1. 左上角状态图标:
- 应用实例的颜色及个数:绿色代表服务正常的实例的数量,黄色代表服务有异常的实例的数量,红色代表服务不可用(宕机)的实例的数量,灰色代表服务未监控的实例的数量。
- 拓扑线的颜色:灰色代表调用服务的平均响应时间小于 200ms,黄色代表调用服务的平均响应时间大于 200ms 小于 1 分钟,红色代表调用服务的平均响应时间大于 1
 分钟。
- 2. 右上角状态图标:
- 纵向展示: 切换中间拓扑图的方向, 如下图是纵向展示:

第87页





点击横向展示仍可切换成横向。

- 拓补图示: 全部图例: 应用: DUBB php 平均访问时延,访问次数 PHP DUBBO JAVA NET 未监控 Ö 8 6 服务: USER LOAD INSTANCES 第三方服务 异步线程 自调用 6 REDIS 数据库: MySQL 0 0 外网应用 8 ORACLE MONGO HSQLDB 2 用户终端 应用 9 12 DB2 P 内网应用 DERBY SYBASE POSTGRE SQL DB2 接收 发送 1 9 ▲ 运行 警告 CASSANDRA DM 4 4 不可用 未监控 消息系统: 🔁 RABBITMQ ROCKETMQ IMS
- 图例:鼠标上移图例,展示对图标的说明。

 列表展示:点击列表展示,会在拓扑图区域分页展示拓扑图中两节点之间的访问 情况列表,包括发起方、被调用方、平均调用时间及响应次数,发起方和被调用 方前面的圆圈颜色代表应用实例的状态,含义与拓扑图上一致,如果没有圆圈则 代表该节点没有状态,同时还可以对发起方和被调用方进行模糊搜索,如下图:



		发起方	Q 被调用方
发起方	被调用方	平均调用时间;	◆ 响应次数 ⇔
End User	 bookdemo@10.10.101.107:8083 	2.89min	9
ASYN#bookdemo@10.10.103.112:4396	10.1.11.175:58082	2.12min	3
bookdemo@10.10.103.112:4396	10.1.11.175:8099	2.12min	3
bookdemo@10.10.103.112:4396	ASYN#bookdemo@10.10.103.112:4396	42.51s	9
eurekaclient@10.10.101.107:8763	 externalservice@10.10.101.107:11111 	10s	88
eurekaclient@10.10.101.107:8763	 externalservicehttps@10.10.103.112:11111 	10s	90
115.236.33.122:18080	 bookdemo@10.10.103.102:8080 	8.98s	27
10.10.103.112	 eurekagateway@10.10.103.112:8765 	6.68s	225
eurekagateway@10.10.103.112:8765	 eurekaclient@10.10.101.107:8763 	6.67s	267
springcloud@10.10.103.112:8765	 springcloud@10.10.101.107:8763 	5.23s	25

3. 中间拓扑图展示



如上图,以横向拓扑图为例,首先是以归并后的结果展示的(应用实例按应用名称合并,非应用实例按类型归并),该拓扑结构图清晰的展现了每个应用分别访问了哪些 服务,每个应用访问各个服务的次数和平均访问延时。如果一项服务中断,就可以立 即看到其它服务遭受的影响。





如上图,鼠标上移到拓扑图的节点上,会高亮展示跟该节点有直接关系的所有数据,同时会以气泡的形式展示该应用的详情:如果是应用的话,会分别展示该应用下正常、异常、宕机三种状态的实例个数及总实例个数;如果是非应用的话,会展示该类型的总个数。

点击某个应用节点会进入该应用的详情页,默认展示该应用下第一个应用实例的数据;当应用展开成实例时,点击可直接跳转到该实例的详情。节点右上角的数字图标表示 归并的实例数量,点击后可展示该节点下的实例。

鼠标上移拓扑图可拖动整个拓扑,上移到节点上也可拖动节点的位置,更方便用户对数据的查看。同时利用鼠标滚轮,可放大缩小整个拓扑图。

4. 右侧节点归并





当拓扑图上通过点击节点右上角的数字展开后,可通过节点归并功能重新归并。打勾表示已经归并,反之表示未归并,可相互切换。

3.4.4. 应用详情

3.4.4.1. 监控概览

该栏目主要展示了某个时间区间访问该应用整体情况,包含:平均执行时间、访问的总次数(吞吐量)、apdex指数和访问该应用的错误率。将鼠标放到图上的某个点可以获得具体的数值同时在图表上展示这段时间内的指标的平均线,使客户可清晰看出超过阈值的时间点以及条数。



列表展示了该实例的事务和数据库事务。



Web事务 ⑦ 切换图					
Name	耗时百分比 👙	apdex 👙	错误率 👙	响应时间 💿 🖕	吞吐量 💠
/webService/webservice	50.37%	0.00	0	2.12min	16.00
/amq/consumerMsg	7.43%	0.00	0	1.00min	5.00
/mongo/search	22.27%	0.35	56.52%	39.15s	23.00
/mongo/insert	vngo/insert 5.94% 0.56 0 26.68s		26.68s	9.00	
/amq/sendMsg	7.43%	0.58	33.33%	25.04s	12.00
/transaction/sleepTime	1.76%	0.17	0	11.84s	6.00
/dubboConsumer/dubboconsumer	0.20%	0.25	50.00%	4.10s	2.00
/transaction/openHrefs	0.27%	0.50	0	2.70s	4.00
/dataBase/mysql/search	2.76%	0.27	0	1.62s	69.00
/redis/insert	0.30%	0.45	45.45%	1.11s	11.00
					< 1 2 3 4 >
数据库事务 ⑦ 切换图					(v)
Name		1	沂属库	sql平均耗时 🖕	吞吐量 👙

3.4.4.2. 应用拓扑

该栏主要以拓扑图的形式展示了某个时间段内与该应用直接关联的服务。拓扑图以查看的应用程序为中心。四周包括了终端用户及应用程序所访问服务。同时还可以得知访问某个服务的次数及平均耗时。此外,还可以拖动图标来改变动态他们的位置。

下图为一周内内以 application@10.10.101.56:9191 为标识的应用的拓扑结构。







.net

应用拓扑的功能基本与 1.3.4.3.拓扑视图 保持一致,增加了对非应用节点的点击功能, 点击后会直接跳转到当前应用实例下的同类型的页面(如点击 mysql 节点,会跳转到该应用实 例的数据库事务页面)。同时不对当前应用的实例做归并。

3.4.4.3. 事务

业务系统所有层级事件的关联,提供面向业务和场景的运维可见度,帮助理解系统行为、 用于分析性能问题的工具,以便发生故障的时候,能够快速定位和解决问题。

该栏目主要展示指定时间段耗时最长的100个Web事务图表及其响应时间和吞吐量,同时还提供了慢事务追踪的功能,让您精准定位慢事务。

WEB 事务页面显示中主要分为三个区块: URL 总览(列表形式展示耗时最长的 100 条 web 事务,可切换成图展示,默认按响应时间倒序排序,也可根据耗时百分比、apdex、错误率和 吞吐量进行排序)、单条 WEB 事务指定时间内的详细信息变化情况(响应时间和吞吐量)、底 部为慢事务追踪详情(指定 WEB 事务指定时间内所有的慢事务,默认按响应时间倒序排序,也 可根据开始时间排序)。



3.4.4.3.1.URL 总览

对应用组件运行的请求事务进行分类、确认访问量较大且响应时间较慢的事务进行优化, 所以要求对监控请求事务在所选时间段内的响应时间和吞吐率进行现场保留及监控。包括 请求事务的名称、耗时百分比、apdex、错误率、响应时间和吞吐量等信息。

应用 apmserver	×				C 7天
事务总览 切換图				根	据事务名称搜索 Q
Name	耗时百分比 👙	apdex 👙	错误率 💠	响应时间 ③ 🖕	吞吐量 ⇔
/webService/webservice	50.34%	0	0	2.12min	16
/amq/consumerMsg	7.42%	0	0	1.00min	5
/mongo/search	22.26%	0.38	54.17%	37.52s	24
/mongo/insert	5.94%	0.56	0	26.68s	9
/amq/sendMsg	7.43%	0.58	33.33%	25.04s	12
/transaction/sleepTime	1.76%	0.17	0	11.84s	6
/dubboConsumer/dubboconsumer	0.20%	0.25	50.00%	4.10s	2
/transaction/openHrefs	0.27%	0.50	0	2.70s	4
/dataBase/mysql/search	2.81%	0.26	0	1.62s	70
/redis/insert	0.30%	0.45	45.45%	1.11s	11
事务总览 切除表格				1	< 1 2 3 4 5 > 根据事务名称搜索 Q
	. 	RL总策	- 8-		

3.4.4.3.2. 单条 WEB 事务的响应时间和吞吐量

对应用组件运行的请求事务详情、确认访问量较大且响应时间较慢的单次事务进行优化, 所以要求对监控请求事务的每个事务的响应时间和吞吐量进行现场保留及监控。包括其中响应 时间包括平均响应时间、服务调用时间(调用外部服务的时间)和数据库响应时间等指标信息。





3.4.4.3.3.慢事务钻取

对应用组件运行的请求事物,监控耗时较长的请求事务,通过慢事务关联分析,能提供代 码级别的可见性以便轻松定位失败点和瓶颈。

点击 URL 总览中的某个事务, 就会在慢事务追踪栏目显示其中最慢的几次事务, 如下图所

示:

• 慢事务追踪列表			
URL名称	开始时间 👙	响应时间 🕜 🖕	操作
/service/device_service/testError	2021-09-03 14:50:49	5.23s	钻取
/service/device_service/testError	2021-09-03 15:02:11	5.23s	钻取
/service/device_service/testError	2021-09-03 14:53:05	5.14s	钻取
/service/device_service/testError	2021-09-03 14:43:59	5.13s	钻取
/service/device_service/testError	2021-09-03 14:55:22	5.13s	钻取
/service/device_service/testError	2021-09-03 14:41:42	5.13s	钻取
/service/device_service/testError	2021-09-03 14:39:26	5.13s	钻取
/service/device_service/testError	2021-09-03 14:37:09	5.13s	钻取
/service/device_service/testError	2021-09-03 14:46:16	5.12s	钻取
/service/device_service/testError	2021-09-03 15:04:28	5.04s	钻取

若想进一步了解慢事务详情,可点击该列表中的慢事务钻取,就可得到此次慢事务钻取后的分析页,如下图所示:



 ・ 一 一 一 一 一 一 一 一 一 一 一 一 一 一 一	持续时间	时间占比	实例信息: newnew 状态码: 200 时间编移量	wnew紀10.10.101.107:8080 消求		
transactionid: 21de5cd5a0ac24e1d9a7b4b01dcf0a282*210 書譯路径 優方法维线 拓扑 自定义方法 系统消息 	持续时间	时间占比	状态码: 200 时间编移量	消求		
調節径 慢方法堆栈 拓扑 自定义方法 系统消息 .apache.tomcat.websocket.server.WsFilter.doFilter	持续时间	时间占比	时间编移量	消求		
.apache.tomcat.websocket.server,WsFilter.doFilter	持续时间	时间占比	时间偏移量	请求		
org.apache.tomcat.websocket.server.WsFilter.doFilter	18.00ms	100.00%	0	{"headers":{"apm- transactionid":"21de5cd5s0ac24e1d9a7b4b01dcf0e262^210","apm-		
org.springframework.web.servlet.FrameworkServlet.service	17.00ms	94.44%	1.00ms	agentid":"newnewnew@10.10.101.107:8090","apm- spanid":"11081696915446723","apm-		
_ javax.servlet.http.HttpServlet.service	17.00ms	Ims 94.44% 1.00ms pspanid":"1","host type":"application	pspanid":"1","host":"10.10.101.107:8080","connection":"keep-alive","content type":"application/x-www-form-urlencoded;charset=UTF=8","cache-			
- com.imooc.appoint.controller.TransactionController.openHrefs	13.00ms	72.22%	3.00ms	control":"no-cache","pragma":"no-cache","accept":"text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2","user-		
java.net.HttpURLConnection.getResponseCode	11.00ms	61.11%	4.00ms	agent":"Java/1.8.0_181"},"response_headers":("set- cookie":"JSESSIONID=48FB1C3E7B82596E8DAD52D76B0A5AF1; Path=/;		
				Http://m/j.1reguest_body-"","response_body-","parama":","utr",/transac ont/ransactor/libeadom", fcooks"1VSESSOND-Ar2A83754C4F326003C1DA5DF8F8A; umrt_uid-asStork-1s00-456-269-dcff64a762505 umrt_uid-asStork-1s00-456-269-dcff64a76275359;_anet_uid_vbit-346 umrt_br_tocids-3483694-8280-46ff-651e-an?6497466; umrt_sched-36847587-1590-4597-368-456614664;		
				请求路径		

钻取页详细展示了 trace 信息,同时多个 tab 栏可查看不同信息。

1) 追踪路径:

在追踪路径点击三角形会显示异常信息及异常代码栈:

追踪路径	慢方法堆栈	拓扑	自定义方法	系统消息							
org.springframe	vork.web.filter.	DncePerReq	uestFilter.doFilter					持续时间	时间占比	时间编移量	请求
_ org.springfra	mework.web.fil	er.OncePer	RequestFilter.doFil	lter				54.33s	100.00%	0	
- org.springt	ramework.web.	ilter.OncePe	rRequestFilter.dof	Filter				54.30s	99.94%	35.00ms	
- org.sprin	gframework.wei	.filter.Oncel	PerRequestFilter.d	loFilter				54.30s	99.94%	35.00ms	
- org.spri	ngframework.w	b.filter.Onc	ePerRequestFilter.	doFilter				54.30s	99.94%	35.00ms	
- org.ap	ache.tomcat.w	bsocket.ser	ver.WsFilter.doFilt	ter				54.29s	99.93%	36.00ms	请求路径
- org.	springframewor	.web.servle	t.FrameworkServle	at.service				54.29s	99.93%	37.00ms	/send
_ jav	 javax.servlet.http.HttpServlet.service 				54.29s	99.93%	37.00ms				
- t	opic.jdbcpool.de	mo.controlk	er.RocketMqContr	oller.sendMsg 📐				54.25s	99.86%	73.00ms	
	org.apache.rocketmq.elient.producer.Default/MQProducer.send 🔤 🛦				4.79s	8.83%	1.44s				
	org.apache.rocketmq.client.producer.DefaultMQProducer.send g 現業详情						7.24s				
	org.apache.roc	ketmq.client	.producer.Default/	MQProducer.send	•			- 1	13.218		
	org.apache.roc	ketmq.client	.producer.Default/	MQProducer.send		 Io.netty.channel.AbstractChannelsAnnotated 	ception	_	18.23s		
	org.apache.roc	ketmq.client	.producer.Default/	MQProducer.send	•	Connection refused: no further information: localhost/127.0.0.1:9		0.0.1:9876	_	23.24s	
	org.apache.roc	ketmq.client	.producer.Default/	MQProducer.send	8	堆栈名	行号			28.26s	
	org.apache.roc	ketmq.client	.producer.Default/	MQProducer.send	8	sun.nlo.ch.SocketChannelimpl	Sacke	etChannelImpl.java:-2		33.27s	
	org.apache.roc	ketmq.client	.producer.Default/	MQProducer.send	8	sun.nio.ch.SocketChannelimpi	Socke	etChannellmpl.java:779		39.29s	
	org.apache.roc	ketmq.client	.producer.Default/	MQProducer.send	8	io.netty.channel.socket.nio.NioSocketChannel NioSocketC		ocketChannel.java:327		44.30s	
	org.apache.roc	ketmq.client	.producer.Default/	MQProducer.send	2	io.netty.channel.nio.AbstractNioChannel\$Abstr actNioUnsafe AbstractNi		actNioChannel.java:340		49.31s	
						io.netty.channel.nio.NioEventLoop	NioEv	ventLoop.java:636			
						> java.net.ConnectException					

2) 慢方法堆栈:展示该事务中该应用的慢方法列表及其堆栈。



追踪路径 慢方法堆栈 拓扑 自定义方法 系统消息

∨ on	g apache.catalina.core.ApplicationFilterChain.internalDoFilter	总耗时:6360ms						
1	org.apache.tomcat.util.threads.TaskThread\$WrappingRunnable.run(TaskThread.java:61))							
2	org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49))							
3	org.apache.tomcat.util.net.NioEndpoint\$SocketProcessor.doRun(NioEndpoint.java:1579))							
4	org.apache.coyote.AbstractProtocol\$ConnectionHandler.process(AbstractProtocol.java:861))							
5	org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:66))							
6	org.apache.coyote.httpl1.Httpl1Processor.service(Httpl1Processor.java:408))							
7	org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:343))							
8	org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:74))							
9) org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:678))							
10	0 org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92))							
11	1 org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:139))							
12	12 org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:526))							
13	org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:96))							
14	org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:202))							
15	org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166))							
16	org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193))							
> or	3.springframework.web.servlet.DispatcherGervlet.doService	总耗时:6180ms						
> or	3.springframework.web.servlet.DispatcherServlet.processDispatchResult	总耗时:4860ms						
> or	springframework.web.servlet.view.AbstractView.render	总耗时:4740ms						
> or	> org.apache.jasper.serviet.JspServietWrapper.service 总相时:4680ms							

(应用中无慢方法的展示历史慢方法数据)

3) 拓扑:

追踪路径 慢方法堆栈 死扑 自定义方法 系统消息			
 正常実例 ● 非常実例 ● 状态码>=400実例 ● 未监控实例 - 平均响应时间<200ms - 200ms<平均响应时间 - 平均响应时间 	URL	发生时间	
	/error	2020-04-21 15:06:21	[放大] 縮小] [纵向展示] [图例]
	/hello-service/scene1	2020-04-21 15:06:20	
	_	关闭	
005.6m, 27. 9pmgov/40.303.02.123.1111 NO70m, 03. Activity/nglov/40.205.00.123.111	117 706.7ms, 1032 Bylegboy@10.100.100	1223/1112 13409.14ms, 728, ASYN#flyingboye910.100.1	0. 123 1110 ^{134,00} Mem, 777 0propose 30. 50. 103 123 1114 2013 7/m, 778 Activative propose 50. 50. 50. 123 1114

从拓扑图中可以清晰的看到此次事务的各部分耗时,让您直观的了解这次事务哪部分耗时 最长。当有安装 netsniffer 时,鼠标上浮在拓扑线上时间展示的地方,会有悬浮框出现,展 示了总体平均时间、纯网络时间及纯网络时间占比等指标。点击拓扑图中除了当前节点外的节 点,当 enduser 调用 A 服务多次时,点击 A 服务时可选择具体事务来查看详情,会跳到新页面 展示该新节点的事务详情页如下图:



UKE A	名称: /transaction/transaction 应用: newcat 敗起 へ 服务响应时间: 8.00ms transactionId: 21de5cd5s0ac24e1d9a7b4b01dcf0e262*210			追踪时间: 2020–0 实例信息: newcaté 状态码: 200	12-29 22:28:39 @10.10.101.107:8080	
追踪路径 慢方法堆栈	a 拓扑 自定义方法 系统消息					
org.apache.tomcat.websocke	at.server.WsFilter.doFilter	持续时间	时间占比	时间偏移量	请求	
 org.apache.tomcat.webso 	ocket.server.WsFilter.doFilter	8.00ms	100.00%	0	{"headers":{"apm- transactionid":"21de5cd5a0ac24e1d9a7b4b01dcf0e262^210"."apm-	
- org.springframework.web.servlet.FrameworkServlet.service		7.00ms	87.50%	1.00ms	agentid":"newnewnew@10.10.101.107:8090","apm- spanid":"11081696915446723","apm-	
_ javax.servlet.http.Http	Servlet.service	7.00ms	87.50%	1.00ms	pspanid":"1","host":"10.10.101.107:8080","connection":"keep-alive","content- type":"application/x-www-form-urlencoded;charset=UTF-8","cache-	
_ com.imooc.appoint.c	controller.TransactionController.search	5.00ms	62.50%	3.00ms	control":"no-cache","pragma":"no-cache","accept":"text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2","user-	
org.apache.tomcat.websocket.server.WsFilter.doFilter org.apache.jasper.servlet.JspServlet.service		4.00ms	50.00%	4.00ms	agent":"Java/1.8.0_181"},"response_headers":("set- cookie":"JSESSIONID=48FB1C3E7B82596E8DAD52D76B0A5AF1; Path=/;	
		4.00ms	50.00%	4.00ms	HttpOnly"},"request_body":"","response_body":"","params":"","url":"/transaction, {"headers":{"cookie":"JSESSIONID=AF42A63754C4F3126003C1DA5D8F8F6A;	
org.apache.jasp	per.runtime.HttpJspBase.service	1.00ms	12.50%	7.00ms	_anaf_uid=ae5b2cfd-b360-4cb9-926b-dfc6f4ea7628; _anaf_hc_added_time=1582880816963;	
					JSESSIONID-81916AF72913CF6035715EBA6D575359;mafuidviait=346; anafhctraceid-34be96b4=280-4dft-61e-ae7696d764ed; anafsos=ba31e287-1299-4b37-8acf-cdef6f1f408de; anafsos=ba31e287-1299-4b37-8acf-cdef6f1f408de; anafbctraceid=34be96b4=320-4b9-009b	
					/transaction/transaction	

4) 自定义方法:

暂无图表

5) 系统消息:

点击系统消息可以查看该次请求消耗的系统资源信息:

追踪路径 慢方法堆栈 拓扑 自定义方法 系统消息

当前CPU使用率	8.17%	当前内存使用率	95.51%
重传率	0.00%	乱序包率	0.00%
∨ 网络连接信息			
LISTEN () ESTABLISHED ()	CLOSE_WAIT ③	TIME_WAIT ③	SYN_RECV ③
22 157	1	85	
∨ 系统状态			
开始时间	CPU时间(ms)	GC时间(ms)	并发用户数
2020-01-09 14:14:11	2854	0	t

3.4.4.4. 异常分析

该模块对应用实例下存在的异常及异常详情进行分析。



3.4.4.4.1.异常列表

下图以列表形式展示了应用实例在所选时间段内发生的异常。可分为代码异常和网络异常,排序展示了异常名称、占比和数量。

异常类型总宽	代码异常	根据类型名称搜索 Q
类型名称	异常占比 🔶	异常数量 👙
java.sql.SQLException	18.76%	109
java.lang.NullPointerException	13.25%	77
com.mchange.v2.resourcepool.CannotAcquireResourceException	10.5%	61
org.springframework.web.util.NestedServletException	8.95%	52
java.lang.ClassCastException	6.88%	40
java.net.SocketTimeoutException	6.2%	36
java.net.ConnectException	6.2%	36
com.mongodb.MongoSocketOpenException	3.27%	19
com.mongodb.MongoTimeoutException	3.27%	19
java.lang.indexOutOfBoundsException	2.93%	17

< 1 2 3 >

3.4.4.4.2.某类型异常数量走势图



3.4.4.4.3.异常详情钻取

点击异常类型的某种异常,就会在异常详细列表显示该异常的每一次发生,如下图所示:



云监控(HC-CloudMonitor)产品说明书

• 异常详情列表				根据事务名称搜索	Q
开始出现时间 🖕	响应时间 📀 👙	事务/异常类型	异常信息	3	操作
2019-11-11 11:29:52	2.17s	/dataBase/mysql/search java.sql.SQLException	Connections could not be acquired from the underlying database!		详情
2019-11-11 11:29:52	2.17s	/dataBase/mysql/search java.sql.SQLException	Connections could not be acquired from the underlying database!		详情
2019-11-10 10:37:12	1.01s	/books/1000/detail java.sql.SQLException	Connections could not be acquired from the underlying database!		详情
2019-11-10 10:19:47	2.13s	/dataBase/mysql/search java.sql.SQLException	Connections could not be acquired from the underlying database!		详情
2019-11-10 10:19:47	2.13s	/dataBase/mysql/search java.sql.SQLException	Connections could not be acquired from the underlying database!		详情
2019-11-10 10:14:52	2.15s	/dataBase/mysql/search java.sql.SQLException	Connections could not be acquired from the underlying database!		详情
2019-11-10 10:14:52	2.15s	/dataBase/mysql/search java.sql.SQLException	Connections could not be acquired from the underlying database!		详情
2019-11-08 20:02:51	817.00ms	/books/1000/detail java.sql.SQLException	Connections could not be acquired from the underlying database!		详情
2019-11-08 20:02:48	156.00ms	/books/1000/detail java.sql.SQLException	Connections could not be acquired from the underlying database!		详情
019-11-08 20:02:38	1.09s	/books/1000/detail java.sql.SQLException	Connections could not be acquired from the underlying database!		详情

若想进一步了解异常详情,可点击该列表中的异常钻取,就可得到此次异常的钻取后分析

页,代码异常和网络异常详情分别如下图所示:

◎ 异常分析				
成起 个	名称: 404 时间: 2020-02-28 11:13:02 transactionId: 5e6543afab26f409abd41005	iddd47ee20^12	实例信息: newcat@10.10.101.107:808 主要影响∪RL: /http/creatHttpStatus	0
 Topo信息 正常实例 • 异常实例 平均响应时间<200ms 	● 状态码>400实例 ● 未监控实例 - 200ms<平均响应时间<1min - 平均响应时间>1i	nin		〔纵向展示〕 [图例]
	End Users	38ms, 1次		

可以看到异常代码、异常栈或拓扑信息。

3.4.4.5. 慢方法

该模块主要展示所选时间段耗时最长的100个方法图表及其响应时间和吞吐量,同时还提供了慢方法调用链,帮助精准定位慢方法。



慢方法页面显示中主要分为三个区块:慢事务总览(图表形式展示耗时最长的100个慢方法,可根据响应时间和吞吐量进行排序)、单个慢方法所选时间段内的详细信息变化情况(响应时间和吞吐量)、底部为慢方法调用链。

3.4.4.5.1. 慢方法总览

慢方法总览 切换图			请输入慢方法名称 Q
Name	响应时间 💿 🝦	吞吐量 💠	操作
com.imooc.appoint.impl.HelloServiceImplService. <init></init>	19.23s	16	钻取
com.rabbitmq.client.impl.SocketFrameHandlerFactory.create	19.22s	10	钻取
com.mongodb.connection.BaseCluster.getDescription	19.21s	19	钻取
org.apache.http.impl.conn.SystemDefaultDnsResolver.resolve	9.96s	1	钻取
com.imooc.appoint.service.lmpl.TransactionServiceImpl.toSleep	9.38s	4	钻取
com.elibaba.dubbo.common.utils.ConfigUtils.getPid	4.80s	1	钻取
com.imooc.appoint.service.Impl.TransactionServiceImpl.openHref	4.77s	2	钻取
om.imooc.appoint.service.lmpl.TransactionServiceImpl\$1.run	4.75s	4	钻取
redis.clients.jedis.Connection.connect	1.48s	7	钻取
org.jboss.netty.channel.DefaultChannelFuture.await0	870.00ms	2	钻取
			< 1 2 >

1. 最慢 top5 方法堆叠

该界面主要展示了所选时间段最慢的5个慢方法的平均响应时间及吞吐量。如下图所示, 清晰的展现了该方法的响应时间和吞吐量。



3.4.4.5.2. 慢方法调用链

选择慢方法总览中的某个方法,点及钻取,就会在慢方法调用链栏目显示该方法的调用过程,如下图所示:





3.4.4.6. 数据库事务

通过对数据库问题的快速定位,对数据库监控要求支持对主流数据库进行运行监测。并能 通过对数据库 SQL 运行情况查看,形成问题记录。

3.4.4.6.1.SQL 总览

对应用组件运行的 SQL 语句进行整体了解,对于应用访问情况进行分析,针对应用组件的 SQL 运行情况。可根据对数据类型和数据库地址进行筛选,以表格的形式分页展示 SQL 语 句的响应时间和吞吐量。表格中的数据默认以 SQL 平均耗时倒序排序,也支持对吞吐量的 排序。如下图



应用 eurekaclient-device v eurekaclient@10.10.101.107:8763 v				① 30分钟
● 全部数据库类型 ∨ 全部数据库地址 ∨				
事务总览 ⑦ 切换图			根据Name搜索	٩
Name	数据库类型	数据库地址	sql平均耗时 🍦	吞吐量 ≑
personal_device#main_device#person_device_relationship/select	jdbc:mysql	10.10.103.112:3306	102.75ms	4
personal_device/select	jdbc:mysql	10.10.103.112:3306	102ms	3
person_device_relationship#personal_device#main_device/select	jdbc:mysql	10.10.103.112:3306	101.73ms	11
personal_device#person_device_relationship/select	jdbc:mysql	10.10.103.112:3306	101ms	4
person_device_relationship/select	jdbc:mysql	10.10.103.112:3306	100.67ms	3
personal_main_device_relationship#main_device/select	jdbc:mysql	10.10.103.112:3306	92ms	1



1. 单条 SQL 事务的响应时间和吞吐率

对应用组件运行的 SQL 语句进行分类、确认访问量较大且响应时间较慢的 SQL 事务进行优化,所以要求对监控 SQL 语句在所选时间段内的响应时间和吞吐率进行现场保留及监控。

点击 SQL 总览中的某条 SQL 语句, 该栏会以图形化的形式展示所选 SQL 语句在所选时间段内的响应时间和吞吐率, 如下图所示:





3.4.4.6.2. 慢 SQL 详情页

SQL名称 库名 追踪时命 sqH时 수 操作	• 慢SQL追踪列表				根据SQL名称搜索		٩
select javatrace interval threshold from analyzer properties where masterin='vanshi sizy dev' and 2020_01_10 17:58:55 751 00ms \$km	SQL名称	库名	追踪时间 💠	sql耗时	÷	操作	
	select javatrace_interval_threshold from analyzer_properties where masterip='yanshi_sjzx_dev'	apm	2020-01-10 17:58:55	751.00ms		钻取	

当存在慢 sql 时会在列表显示,点击钻取后可展示详情,如下:

> 慢SQL分	分析	
		耗财 751.00ms 数据库类型: mysql 库名: apm
详细SQL	L语句	
select jav	watrace_interval_threshold from analyzer_properties where masterip='yanshi_sjzx_dev	/
✓ trace信息	息	
1 com.n	.mchange.v2.c3p0.impl.NewProxyPreparedStatement.executeQuery(NewPro	oxyPreparedStatement.java:79)
2 sun.r	.reflect.NativeMethodAccessorImpl.invoke0(Native Method)	
3 sun.r	reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.	java:62)
4 sun.r	, reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAcces	sorImpl.java:43)
5 java.	.lang.reflect.Method.invoke(Method.java:498)	
6 org.s	springframework.web.method.support.InvocableHandlerMethod.doInvoke	e(InvocableHandlerMethod.java:221)
7 org.s	springframework.web.method.support.InvocableHandlerMethod.invokeFo	orRequest(InvocableHandlerMethod.java:136)
8 org.s	springframework.web.servlet.mvc.method.annotation.ServletInvocable	eHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:114)
9 org.s	.springframework.web.servlet.mvc.method.annotation.RequestMappingHa	andlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:827)
10 org.s	.springframework.web.servlet.mvc.method.annotation.RequestMappingHa	andlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:738)
11 org.s	.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapte	er.handle(AbstractHandlerMethodAdapter.java:85)
12 org.s	.springframework.web.servlet.DispatcherServlet.doDispatch(Dispatch	erServlet.java:963)
13 org.s	.springframework.web.servlet.DispatcherServlet.doService(Dispatcher	rServlet.java:897)
14 org.s	springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.processRequest)	workServlet.java:970)
15 org.s	.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServle	et.java:861)
16 java>	<pre>ax.servlet.http.HttpServlet.service(HttpServlet.java:687)</pre>	
17 org.s	.springframework.web.servlet.FrameworkServlet.service(FrameworkServ	vlet.java:846)
40 20000		

可展示该条 sql 的基础信息、详细 SQL 语句和 trace 信息。

3.4.4.7. 数据库连接池



数据库连接池的运行情况对整个应该服务的运行起到了非常关键的支撑作用。对其监控就 显得非常有必要了。通过监控某时间段内连接、可用连接数、活跃连接数和连接池最大连接数 等获取数据库连接池情况,快速定位数据库连接池情况。

数据库连接池总览

数据库连接池总宽 ②						根据名称,类型,地址搜索	٩
数据源名称	数据源类型	jdbc地址	应用容器类型	可用连接数 👙	活跃连接数 👙	连接池最大连接数 👙	
2sajxsa7xxdcuufmbny 22f057b4	C3p0	jdbc:mysql://10.10.103.112:3306/	tomcat	10	0	20	

如上图,可展示该时间段内连接池最新一次数据的数据源名称、数据源类型、jdbc 地址、 应用容器类型、可用连接数、活跃连接数和连接池最大连接数。

点击表格中的某一条数据,可在下方展示其详情。

3.4.4.7.1.数据源详情

主要展示该数据源时间段内的历史使用信息和历史使用趋势图。

2sajxsa7xxdcuufmbny	22f057b4								
• 历史使用信息 ②		• 历史使用趋势	N						
						- 可用连接数 -〇- 最	大连接数		
		20个 ₁							·p
最大连接数	20个								
活跃连接数	0	15个							
活跃连接数占比	0	10个						,	
可用连接数	10个							/	/
可用连接数占比	50.00%	5个							<i>k</i>
		0 0	01-08	01-09	01–10	01–11	01–12	01-13	01–14

3.4.4.8. NoSQL

在目前大型服务应用中,非关系型数据库的使用也越来越频繁,对非关系数据库的监控也越发重要,特别是对慢 nosql 的监控。



3.4.4.8.1.nosql 总览

■ 全部数据库类型 ∨	全部数据库地址	\vee			
NOSQL总览 ⑦ 切换图				根据Name搜索	٩
Name	数据库类型	数据库地址	nosql平均耗时 🖕	吞吐量 ⇔	
mongoTest.mongoTest/find	mongo	47.100.17.213:27017	112ms	4	
mongoTest.mongoTest/insert	mongo	47.100.17.213:27017	111ms	2	

如上图,可展示 mango、red is 和 cassandra 三种类型的 nosql,看到平均耗时和吞吐量。

可切换为图表展示:



3.4.4.8.2.单条 NoSQL 事务的响应时间和吞吐率

该模块主要展示了所选时间段内某条 nosql 的平均响应时间及吞吐率。

如下图所示,清晰的展现了每个事务的响应时间和吞吐率。





3.4.4.8.3.慢 nosql 详情页

• 慢NOSQL追踪列表	根据SQL搜	根据SQL搜索 Q	
Name	开始时间 🖕	平均耗时 👙	操作
Operation:Find;Filter;{ "age" : { "\$numberLong" : "28" } };Modifiers:null;Projection:null;Sort:null@amp_testdb.mongoTest	2019-11-11 11:30:37	2.00ms	钻取
Operation:Find;Filter:{ "name" : "value08" };Modifiers:null;Projection:null;Sort:null@amp_testdb.mongoTest	2019-11-08 17:39:10	2.00ms	钻取

当存在慢 nosql 时会在列表显示,点击钻取后可展示详情,如下:

○ NOSQL分析								
	レビン 收起 へ	名称: mongoTest.mongoTest/insert 应用: newnewnew nosql耗时: 133.00ms 数据库类型: mongo	道歸时间: 2020-02-28 17:26:33 实例信息: newnewnew@10.10.101.107:8090 数据库地址: 47.100.17.213:27017					
∄详 Op	田SQL语句 eration:Insert:Ducur	ment:{ "ace" : "100", " id" : { "\$oid" : "5e58dcc99f09740c	:372bf7cf" } }:Filter.null@monaoTest.monaoTest					
🤊 tra	ce信息							
1 com.mongodb.Mongo.execute(Mongo.java:749)								
<pre>2 com.mongodb.Nongo\$2.execute(Mongo.java:730)</pre>								
3	<pre>3 com.mongodb.MongoCollectionImpl.executeSingleWriteRequest(MongoCollectionImpl.java:482)</pre>							
4	<pre>4 com.mongodb.MongoCollectionImpl.insertOne(MongoCollectionImpl.java:277)</pre>							
5	com.imooc.appoint.service.Impl.MongoServiceImpl.Insert(MongoServiceImpl.java:34)							
	com.imooc.appoi	int.controller.MongoController.insert(MongoCo	ontroller.java:45)					

3.4.4.9. 服务调用

应用的URL调用监控为提升应用服务质量提供了非常必要的数据依托。

3.4.4.9.1. 服务调用总览

服务调用总览 ⑦		全部调用變型	~	全部协议	类型 > 相	据名称搜索	٩
名称	调用类型	全部调用类型		軽 💠	响应时间 🕐 💠	吞吐量 ⇔	操作
http://10.1.11.175:8099/hello	内部调用	外部调用			2.12min	2	查看
http://10.1.11.175:58082/jeeshopclient	内部调用	内部调用			2.12min	2	查看
http://jeeshopclient.cdev.whchem.com	外部调用	http	0		166.25ms	4	查看

该列表展示了所有了调用类型(内部、外部),展示了调用的错误率、吞吐量和响应时间, 有助于分析服务质量。



3.4.4.9.2.调用情况汇总图



该图表展示了平均响应时间最久和错误率最高的服务调用曲线图

3.4.4.9.3. 服务调用详情



在服务调用列表中点击查看可显示此条调用服务的详情,响应时间、吞吐率;若本次 调用为慢服务,会对其开始时间、响应时间及 trace 信息进行分析。


3.4.4.10. 消息队列

当消息队列里的消息出现堆积时,我们排查的方式往往时比较盲目的。对消息队列及队列 里的消息进行监控数据就成为了排查的新方向。

3.4.4.10.1. MQ 一览

应用 bookdemo	实例 ✓ ● bookdemo@10.10	.103.112:4396	v		C 30分钟
生产者 消费者					
MQ一览 ⑦				全部MQ类型 >	根据Name搜索 Q
Name	MQ类型	消息总数 ⇔	吞吐量 ⇔	平均消息发送时间 🖕	慢消息比例 ↓
47.100.17.213:5672	rabbitmq	10	10	0	0

如上图,可以生产者和消费者的不同角度展示消息队列的地址、消息总数、每分钟消息数和平均消息发送时间(消费者为平均消息处理时间),整体把握消息队列的健康情况

3.4.4.10.2. 单个消息队列的详情



如上图,主要展示了该选择时间段内的平均消息数和平均消息发送时间的趋势,及慢消息

追踪列表。

点击钻取一条具体的慢消息追踪,展示详细的慢消息分析信息。

第 109 页

◎ 慢消息分析								
	名称: today 时间: 2020-02-28 17:26:47 实例信息: newnewn@10.10.101.107:8090 transactionld: 21de5cd5a0ac24e1d9a7b4b01dcf0e262^171	地址: 47.100.17.213:5672 MQ类型: rabbitmq						
♪ Topo信息 ● 正常安例 ● 异常安例 ● 状态码>400实例 ● 未監控实例 ● 平均响应时间<200ms - 200ms<平均响应时间<1min - 平均响应时间>1min								
	B							

3.4.4.11. JVM 概况

当应用出现异常如 oom 时,往往不能十分准确得获取到发生异常时 JVM 的内存使用情况的信息。

该模块主要展示了系统参数、JVM 堆内存各区域内存使用情况、线程信息、垃圾收集及类 加载等相关信息。

3.4.4.11.1. 系统参数和内存池

下图左侧展示了系统的主要参数,其中包括 java 运行时的环境版本、使用的 JIT 编译器名称、用户主目录等信息。



应用 bookdemo	<u>実</u> 例 ● bookdemo@10.10.101.107:8083 ∨	① 30分钟
系统参数 ⑦ awt.toolkit catalina.base catalina.bome	sun.awt.X11.XToolkit /home/zanebono/apache-tomcat-8.5.35 /home/zanebono/apache-tomcat-8.5.35	内存池 ⑦ MermyoryPool v
catalina.useNaming client.logFileMaxIndex	true 10	13.976
client.logLevel client.logRoot	INFO /root/logs/rocketmqlogs	8.386
common.loader file.encoding	"\${catalina.base}/lib","\${catalina.base}/l UTF-8	5.59G
file.encoding.pkg file.separator	sun.io /	•
hc_pid ignore endorsed dirs	144841	-2.79G J CodeCa- PSEden- PSSurv- Metasp- Compre- PSOldG- che Space IvorSpace ace ssedClassSpace en

栏目右侧展示了 JVM 堆内存的使用情况,包括 Eden、Survior、OldGen 及 PermGen 等 区域。用曲线图分别展示了这段时间内每个区域的初始内存大小(init)、当前使用的内 存总量(used)、可使用的内存量(commited)、可用的最大内存量(max)。此外将鼠 标放到曲线图上还可以查看具体的数值。

3.4.4.11.2. 线程信息

栏目中间部分展示线程的信息,从左至右依次表示当前线程数,死锁线程数、守护线程数, 程序运行过程中历史最大存活线程数及累计使用的线程数。在下面的窗口中分页展示了每个线 程的详细信息,包括线程名称、线程的 ID、CPU 占用率、线程状态等信息,当某个线程处于 WAITING 状态时,可以看到当前线程所等待锁的名称。此外,点击某个线程后,可以从右侧获 得该线程更加详细的信息。

点击死锁线程 tab,下方列表变更为死锁线程的详情展示。



线程名称 ⇔	线程ID ≑	CPU占用率 👙	线程状态 💲	锁对象 💠	线程: main	
main	1	0%	RUNNABLE	none	threadId	1
Reference Handler	2	0%	WAITING	java.lang.ref.Reference\$Lock@50606818	lockOwnerld	-1
Finalizer	3	0%	WAITING	java.lang.ref.ReferenceQueue\$Lock@7478aca4	threadState	RUNNABLE
Signal Dispatcher	4	0%	RUNNABLE	none	cpuPercentage	0
AsyncLogger-1	6	0%	WAITING	java.util.concurrent.locks.AbstractQueuedSync	stackTrace	
AsyncFileHandlerWriter-1	9	0%	TIMED_WAITING	java.util.concurrent.locks.AbstractQueuedSync	lockName	none
reaper-1	10	0%	RUNNABLE	none	threadName	main
	44	0%	DUNNARI E			
othread-2		070	HONNADEL	none		
othread-2 Apm Sender	12	0%	TIMED_WAITING	none		
othread-2 Apm Sender eaper-1 线程信息	12 13	0%	TIMED_WAITING RUNNABLE	none none <12345185		
iothread-2 Apm Sender reaper-1 线程信息 3 死锁线程数	12 13 ⑦ 17	0% 0% 2 当前线程	TIMED_WAITING RUNNABLE	none none <1 2 3 4 5 ···· 18 > 守护线程数 ⑦ 173 历史最大线	程数 ⑦ 217	累计使用线程数 ④
iothread-2 Apm Sender reaper-1 线程信息 3 死锁线程数 程名称 (已锁线程) 0	12 13 ⑦ 【17 线程ID ↓	0% 0% 0% 2 当前线程 CPU占用事 ◆	timed_wanting RUNNABLE 数 ⑦ 171	none none c 1 2 3 4 5 ···· 18 > 守护线程数 ⑦ 173 历史最大线 顿对象 \$	程数 ⑦ 217 4程: main	累计使用线程数 ③
othread-2 Apm Sender reaper-1 线程信息 3 死锁线程数 程名称(已锁线程) \$	12 13 ⑦ 17 线程ID ↓ 135	0% 0% 0% 2 当前线程 CPU占用率 ◆ 0%	TIMED_WAITING RUNNABLE 数 ⑦ 171 线程状态 ÷ DEADLOCK	none none none <1234518、	程数 ⑦ 217 线程: main threadId	累计使用线程数 (2) 135
tothread-2 Apm Sender eaper-1 战程信息 【 名 死锁线程数 章 程名称(已锁线程)章 程1	● 12 13 ● 13 17 鉄程ID ↔ 135 136	2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3	timed_waiting RUNNABLE timed_waiting RUNNABLE timed_waiting t	none none none <1234518、	程数 ① 217 线程: main threadId lockOwnerId	累计使用线程数 (135 136
othread-2 Apm Sender reaper-1 线程信息	● 12 13 13 13 13 5 135 136 233	2 2 2 2 2 3 3 3 3 4 3 4 5 7 4 5 5 7 4 5 5 7 7 7 7 7 7 7 7 7 7 7 7 7	timed_waiting RUNNABLE	none none none <1234518、	程数 ③ 217 线程:main threadId lockOwnerId lockedMonitors	累计使用线程数 ① 135 136 java.lang.Object
tothread-2 Apm Sender eaper-1 名 変で锁线程数 全名称(已锁线程) 章 程1 定 2	12 13 ○ 17 <u>鉄程</u> D ↓ 135 136 233	2 当前线程 CPU占用車 ◆ 0% 0% 0%	TIMED_WAITING RUNNABLE 数 ② 171 线程状态 章 DEADLOCK DEADLOCK	none none none ⑦中珍线程数 ⑦⑦中珍线程数 ⑦173 历史最大线[調对象 \$java.lang.Object@32a31ed2java.lang.Object@32a31ed2	程数 ③ 217 线程: main threadId lockodMonitors threadState any@competing	累计使用线程数 ④ 135 136 java.lang.Object DEADLOCK o
tothread-2 Apm Sender eaper-1 3 死锁线程数 程名称(已锁线程) © 程1 章 程2	12 13 ○ 17 <u>线和</u> D ↓ 135 136 233	2 当前线程 CPU占用車 ◆ 0% 0% 0%	TIMED_WAITING RUNNABLE 数 ② 171 续程状态 章 DEADLOCK DEADLOCK	none none none <(1)2345…185 (可护线程数) (打了3 历史最大线 词对象 章 java.lang.Object@32a31ed2 java.lang.Object@32a31ed2	程数 ⑦ 217 线程: main threadId lock@MonItors threadState cpuPercentage stackTrace	累计使用线程数 (135 136 java.lang.Object. DEADLOCK 0 waiting for loc
tothread-2 Apm Sender eaper-1 3 死锁线程数 程名称 (已锁线程) 0 程1 程2	12 13 つ 13 13 13 13 135 136 233	2 当前线程 CPU占用車 ↓ 0% 0% 0%	TIMED_WAITING RUNNABLE 数 ② 171 線程状态 章 DEADLOCK DEADLOCK DEADLOCK	none none none <12345…185 (可於経程数 ⑦ 173 历史最大线 領対象 章 java.lang.Object@32a31ed2 java.lang.Object@32a31ed2	程数 ⑦ 217 线程: main threadId lockOwnerId lockedMonitors threadState cpuPercentage jstackTrace lockName	累计使用线程数(135 136 java.lang.Object. DEADLOCK 0 waiting for loc java.lang.Object.

3.4.4.11.3. GC 详情

如下图所示,通过GC详情可以得知JVM使用了哪些垃圾收集器,同时还可以得到每 个垃圾收集器执行的次数和总时间。





3.4.4.12. 线程剖析

当出现死锁等线程阻塞问题时,往往会导致整个应该无法正常工作,这时候就需要用类似 jstack 来分析线程状态。然而对于不熟悉 jstack 的人员来说,其提供的信息晦涩而难以理解。 平台此模块为用户提供了下载线程信息并剖析 jstack 信息的功能。

应用 newnewnew v 冬例	ewnew@10.10.101.107:8090			
1 线程剖析可按照用户所需时间间隔、次数	α,控制agent去获取应用线程信息,等待	间隔时间后,便可进行下载分析。		
*剖析时间: 2020-03-03 13:40:39 📋	* 剖析时长: 请输入时长	(单位秒) * 剖析次数	筑: 请输入次数	开始剖析
线程剖析列表				
剖析时间	剖析时长	剖析次数	状态	操作
2020-02-21 14:03:54	10s	1次	已完成	下载 删除
2020-02-21 14:11:46	20s	2次	已完成	下载 删除
2020-02-28 11:09:05	1s	1次	已完成	下载 删除
2020-02-28 12:09:05	1s	1次	已完成	下载 删除
2020-03-02 14:38:49	1s	1次	已完成	下载 删除
● 正常 ● 异常 ● 状态码>400 ● 未始 - 平均响应时间<200ms - 200ms<平封 End User<10.100.100.123>	5 5 9 9 9 9 9 9 9 9 9 9 9 9 9	Imin Oms, 1% 101.107.8090 Ims, 1%	10.10.101.107:20880	

3.5. 告警

目前集群所承载的业务越来越多,集群的复杂程度也越来越高,发生故障及异常的频率也 水涨船高。构建一套告警系统逐渐提升至第一优先级,及时有效的告警信息往往能在预防集群 第113页



异常减小隐患方面起到举足轻重的作用。云监控平台整合多种来源资源数据,结合用户自定义 指标配置,通过智能算法,从杂乱无序的异常信息提取出有效的信息归并成告警信息展示并通 过邮件短信等方式及时通知相关人员。

告警模块通过多个纬度对告警事件进行查询及配置:

3.5.1. 告警事件

支持 pod 异常告警、主机节点异常告警、工作负载异常告警、应用异常告警等多种告警,可快速定位告警原因,解决告警问题。

告警原因:	创建时间: 开始日	明 ~ 结束日	相				重置	查询
	告警内容	告警等级 👙	告警对象	告警类型 🔻	告警时间 💠	通知对象	状态 🔻	操作
	\${Cluster} 集群下\${namespace}命名空间下nephele-0pod pod 状态异常 当监控值持续2分钟,当前值9919	P2	nephele-0	P Pod告警	2021-12-23 11:36:00	R DIO	。待处理	处理
	\${cluster} 集群下\${namespace}命名空间下nephele-0pod pod 状态异常 当监控值持续2分钟,当前值9908	P2	nephele-0	 □ 工作负载告警 Pi □ 应用告警 □ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	2021-12-23 11:25:01	R DIO	。待处理	处理
	\${cluster} 集群下\${namespace}命名空间下nephele-0pod pod 状态异常 当监控值持续2分钟,当前值9898	P2	nephele-0	□ 组件告誓 P(确定 重置	2021-12-23 11:15:00	R DIO	。待处理	处理
	\$(cluster) 集群下\$(namespace)命名空间下myjmeter- 845b58b5b9-6pbn9pod pod异常 当监控值持续2分钟, 信 息:\$(message)	P1	myjmeter-845b58b5b9- 6pbn9	Pod告警	2021-12-22 18:56:00	R, testGroup1	• 待处理	处理
	\${cluster} 集群下\${namespace}命名空间下nginx-ingress- controller-9649f46d-rmf7gpod pod异常 当监控值持续2分 钟,信息:\${message}	P1	nginx-ingress-controller- 9649f46d-rmf7g	Pod告警	2021-12-22 18:56:00	R testGroup1	• 待处理	处理
	\${cluster} 集群下\${namespace}命名空间下nginx-ingress- controller-9649146d-rm17gpod pod异常当监控值持续2分 钟,信息:\${message}	P1	nginx-ingress-controller- 9649f46d-rmf7g	Pod告警	2021-12-22 18:56:00	R testGroup1	• 待处理	处理
	\${cluster} 集群下\${namespace}命名空间下bookdemo- 7d8f8b7f96-ng247pod pod异常 当這控值持续2分钟,信 息:\${message}	P1	bookdemo-7d8f8b7f96- ngz47	Pod告警	2021-12-22 18:56:00	R testGroup1	• 待处理	处理
	\${cluster} 集群下\${namespace}命名空间下strimzi-cluster- operator-6ccbcb96f6-gnvdrpod pod异常 当监控值持续2分 钟, 信息:\${message}	P1	strimzi-cluster-operator- 6ccbcb95f5-gnvdr	Pod告警	2021-12-22 18:56:00	R testGroup1	◎ 待处理	处理
	\${cluster} 集群下\${namespace}命名空间下nephele-0pod pod 异常 当监控值持续2分钟,信息:\${message}	P1	nephele-0	Pod告營	2021-12-22 18:55:00	R testGroup1	。待处理	处理

3.5.2. 告警规则

可根据实际情况设置告警的规则,单系统匹配到设置的告警规则时,将触发告警信息的发送, 及时提示用户。



3.5.2.1. 告警规则列表

列表展示了所设置的所有告警规则,包括告警规则名称、告警类型、监控指标及通知对象。

如下图:

告警事件	名称:		创建时间: 开始日期 ~ 结束日期 白				重置 重询
告警规则							
通知对象	新建规则						
通知对象组	名称	告警类型	监控指标	创建时间	通知对象	状态 👻	操作
	node test	主机告警	cpu使用率(%)	2021-12-28 17:15:48	R JOJO	〇已停用	编辑删除
	节点磁盘	主机告警	磁盘使用率(%)	2021-12-27 16:47:41	& Jonny	已停用	编辑剧除
	虚机请求字节数告警	应用告警	请求字节数	2021-12-27 16:46:37	∞ 杨文宣	〇已停用	编辑剧除
	虚机错误率告警	应用告警	请求错误率	2021-12-27 15:48:59	∞ 杨文宣		编辑剧除
	虚机内存使用量告警	主机告警	内存使用率(%)	2021-12-27 15:25:40	_ 局 杨文宣	日序用	编辑删除
	nodeload1	主机告警	一分钟负载	2021-12-25 17:52:32	R DIO	〇已停用	编辑剧除
	pod mem usage	Pod告警	内存使用率(%)	2021-12-25 17:51:57	R DIO	〇日停用	编辑删除
	pod cpu usage	Pod告警	cpu使用率(%)	2021-12-25 17:51:28	R DIO	日常用	编辑删除
	node 内存使用率	主机告警	内存使用率(%)	2021-12-25 17:36:41	8 杨文宣	〇已停用	编辑删除
	node cpu使用率	主机告警	cpu使用率(%)	2021-12-25 15:52:09	A 杨文宣	〇日停用	编辑删除
				第 1-10 争/总井	474 1 2 3	4 5	10条/页 🗸

3.5.2.2. 告警规则详情

点击某条告警规则,可查看此条告警规则的详情:



告警内容: \${cluster} 集群下 \${alertobject}主机 \${alertmetric} \${trigger},当前值\${metricvalue}

状态:已停用

告警等级: P4

告警指标: cpu使用率(%)

② 规则详情

删除 编辑

规则基本信息 名称: node test

创建时间: 2021-12-28 17:15:48

告警类型:节点告警

压缩时间:20

通知对象:

A JOJO A testGroup1

规则条件

节点名称 不包含 d

触发条件

单次

大于 1

3.5.2.3. 新建告警规则

点击新建告警按钮,可进行告警规则的新建。

1 选择告警检测对象	2 配置触发条件	3 设置告警内容与通知对象
*告警名称:		
压缩时间:	5 分钟	
* 告警指标:		V
条件全部满足时生成告警:	×)	
	+ 增加条件	

第 116 页



通过三个步骤:选择告警检查对象 -> 设置触发条件 -> 设置告警内容与通知对象之后即 可创建新的告警规则。

3.5.3. 通知对象

3.5.3.1. 通知对象列表

通知对象列表展示了所有的通知对象:

新建对象					
名称	所属通知组	邮箱	手机	创建时间 💠	操作
notifyUser1	R testGroup1	163@163.com	17899999999	2021-10-18 16:07:25	编辑 删除
Jonny	R DIO R JOJO	Joestar7@110.com	17899999999	2021-10-22 17:01:48	编辑 删除
Gyro	PR DIO	Zeppelin@7.con	17899999999	2021-10-22 17:03:13	编辑 删除
杨文宣		ywx@harmonycloud.cn	17361893733	2021-12-22 17:08:20	编辑 删除

3.5.3.2. 通知对象新建

点击新建通知对象按钮,可跳转新建通知对象界面,输入相关通知对象信息即可新建 一个通知对象。在通知规则中配置该通知对象之后,发生对应异常告警时,该通知对象即 可收到相应通知。



请输入名称	创建时间:	^{开给日期} 新建通知对象	東日期 円	×	
主 对象		* 名称:			
ĵ۲	所属通知组	* 邮箱 ·			创建时间 💠
ifyUser1	R testGroup1				2021-10-18 16:07:25
ากy	R DIO R JOJO	* 手机号:			2021-10-22 17:01:48
ro	Ph DIO	所属通知组:			2021-10-22 17:03:13
之宣				取消 确定	2021-12-22 17:08:20
				×.	第 1-

3.5.4. 通知对象组

在某些场合,一个告警通知发生需要通知一组人员,那么这个时候将这组人员划分为一个 对象组是一个合理的选择。

3.5.4.1. 通知对象组列表

新建对象组			
名称	通知对象数 😄	创建时间 🗘	操作
testGroup1	1	2021-10-18 16:07:41	编辑删除
OLOL	1	2021-10-22 17:01:59	编辑 删除
DIO	2	2021-12-03 10:06:56	编辑 删除

第 1-3 条/总共 3条 < 1 > 10 条/页 >

3.5.4.1.1. 通知对象组新建

点击新建通知对象组按钮,可跳转至新建通知对象组界面:



创建时间:		开始日期 ~	结束日期	T=1			
		新建通知对象组				×	
		* 2称·				7	
	通知对	E 197					
		添加通知对象:					
	1					I	
	1				取消	确定	
	2				2021-12-	03 10:06:56	

可选择该通知对象组所需要包含的通知对象,保存之后即可创建一个通知对象组。当在告 警规则中配置了该通知对象组之后,对应发生的告警即可通知到该对象组下面的所有对象人员。

3.6. 仪表盘

仪表盘主要是对容器的监控,此模块对容器的各种指标进行了详细且具体的展示,加上图 表形式的表现方式,是容器状态与指标变得一目了然。

3.6.1. 面板列表



ofault > 容器云集群看板 →		
1.instance 10.42.7.96:8080 - 高性能磁盘 All - node nwcsy2-07 -		
> 高性能集群data使用情况 (1 panel)		
> 容量监控&容量预估 (12 panels)		
> 集群统计看板 (3 panels)		
> 集群详情看板 (10 panels)		
∽ deployment详情看板		
容器云集群看板 -	💵 🖈 🖻 🛱 🖵 🥥 Last 10) minutes - Q 2 10s -
		1
1.instance 192.168.32.70:8443 ▼ 高性能磁盘 All ▼ node None ▼		
> 高性能集群data使用情况 (1 panel)		8



如上,默认呈现容器云集群 dashboard,查看当前集群下的高性能 data 使用情况、容器监控&容量评估、集群统计看板、集群详情看板、deployment 详情看板等信息,分析当前集群的整体情况。

右上角呈现操作控件,包含添加面板、标记为收藏夹、分享仪表盘、保存仪表盘、仪表盘设置、 循环视图模式以及时间控件等,实现对当前 dashboard 的操作。

3.6.2. 创建仪表盘



点击创建仪表盘可跳转如下页面:

📰 New dashboard 🗕			
■■■●● 新建面板			×
\$	》 添加 查询	选择 可视化	
	转	奂为行	

点击查询查看此仪表盘的一系列数据:



点击可视化:





如上图,通过添加查询以及选择可视化方式,实现对 dashboard 图表的添加,选择一个数据源,在"查询"选项卡的第一行中,单击下拉列表以查看所有可用的数据源。在"面板"选项卡的"可视化"部分中,单击一种可视化类型,保存仪表盘。

3.6.3. 创建文件夹

按需创建文件夹,并根据角色权限,分配所需的文件夹,提供文件夹内的 dashboard 视图。如下图:

	□□ () () () () () () () () () () () () () () () () ()
	▲ 管理 局 播放列表 ● 快照
	New Dashboard Folder
	Name
la de la companya de	Create

3.6.4. 导入



通过 Upload .json file 按钮,导入本地 yaml 文件,实现 dashboard 的创建。

导入 Import dashboard from file or Grafana.com		
Grafana.com Dashboard	Upload .json file	
Paste Grafana.com dasnboard un or id		
ELoad		

3.6.5. 添加数据源

能置 组织: yanshi			
🗢 数据源 💄 用户 💄 团队	. 🖌 插件 🛛 莘 偏好设置	💊 API 密钥	
Q Filter by name or type			添加数据源
Prometheus http://prometheus-k8s.monitoring:909	90		PROMETHEUS
Prometheus-1 http://prometheus-k8s.monitoring:909	30		PROMETHEUS

提供不同数据源下的支持能力,包含时序数据库、日志&文档数据库、SQL、云插件、企业插件



等数据源能力。点击添加数据源,可选择对应的数据源,导入数据,以Prometheus数据举例, 点击Prometheus,进入Prometheus 配置界面如下图:

Data S Type: Prom	Data Sources / Prometheus									
⊉ Settings I	Dashboard	ls								
Name 🚯	Prometheus			Default						
НТТР										
URL ®	http://prom	etheus-k8s.monitori	ng:9090							
Access	Server (defa	ault)		Help >						
Whitelisted Cookies ④			Add							
Auth										
Basic auth		With Credentials								
TLS Client Auth	O	With CA Cert								
Skip TLS Verify	0									
Forward OAuth Identity	0									
Custom HTTP Headers										
Add Header										
Scrape interval										
Query timeout										
HTTP Method										
Mise										

需配置数据源名称、Prometheus 服务器的 URL、默认服务器、对 Prometheus 数据源启用 基本身份验证、基本身份验证的用户名、基本身份验证的密码等信息,完成 Prometheus 数据 源的支持网络监控。完成数据源配置后,在添加面板时,即可选用当前数据源作为 dashboard 面板的数据源,实现可视化配置及呈现。

3.7. 网络

3.7.1. DNS 异常



当网络请求异常时,第一步往往偏向于排查 DNS 通信是否正常。平台通过对 DNS 请求进行 监控并且对 DNS 异常信息进行收集优化,通过统一模块页面对异常 DNS 数据进行展示。

3.7.1.1. DNS 异常概览

在 DNS 概览页面,可通过查询响应时间、请求数、错误数等排查出网络请求 DNS 异常的汇总数据,可通过对源段、目的端等继续筛选进而更加明确细化异常信息。



3.7.1.2. DNS 异常详情

点击某次请求可进入查看此次请求的异常详情页面:



DNS详情

			◎ 目的端	
源服务			目标服务	kube-dns
源IP	192.168.130.67	连接成功(错误请求)	目标IP	10.10.102.161
源集群	k8-159-rel	\longrightarrow	目标集群	k8-159-rel
源节点	159master		目标节点	161node
源Pod	cattle-cluster-agent-57847 756b4-xwdmx		目标Pod	
成名:git.rancher.id	o.cattle-system.svc.cluster.local.	请求时间:20	021-12-30 11:14:54	
ī求耗时:1.06μs		响应耗时:63	22.15µs	
回码: NXDomai	n	连接时间:0	ns	

DNS 详情页面展示了此次 DNS 异常请求的一些重要指标及状态,包含了源段指标与目的端指标及此次请求耗时、相应耗时、返回码及连接耗时。

3.8.检索

当发生故障后,日志系统对与复现历史现场有着无与伦比的重要性。平台提供多种日志收集方案对日志进行收集,并对海量日志进行有效性检索,以便能对故障现场进行快速地位。

3.8.1. 日志收集

1. 平台日志收集

平台日志的收集可以通过日志采集 agent 来实现。在平台所管理的所有主机上部署日志采 集 agent,首先对日志进行本地化存储。日志采集 agent 可以在本机提供日志缓冲和压缩能力, 降低对服务端的日志传输压力。日志的服务端入口使用 Kafka 将日志的摄入和处理解耦,降低 日志摄入的延迟并提升日志处理的可扩展性。日志存储方面采用分布式存储的方案以保证日志 的完整性和安全性,所有日志存储到高可用、可扩展的 Elasticsearch 集群中。当遇到网络问

第 126 页



题,日志无法传输时,日志采集 agent 会在本地保留日志并重试传输;即便服务端失效造成日 志无法传输,日志仍然可以保留在本地并可以进行离线处理。



平台日志收集示意图

3.8.2. 事件检索

对平台发生的事件进行查询,可输入有效的筛选条件进行筛选。查询出的事件日志以列表 的方式进行展示。列表内容包括:事件详情、事件原因、命名空间、事件状态、事件所属集群、 事件发生时间等有效信息。如下图:



云监控(HC-CloudMonitor)产品说明书

租户: zzzzzzz 事件状态: 正常	×	k8s集群: 对象名称:	vm-50-test	V		命名空间: 事件详情:	cloudmonitor
事件原因:		对象时间:	2021-12-23 13:43:32 ~ 202	21-12-30 13:48:32			重置 查询
事件详情	事件原因	命名空间	事件状态	k8s集群	对象类型	对象名称	事件时间 🗘
Started container mysql	Started	cloudmonitor	 正常 	vm-50-test	Pod	mysql-mm-sts-0	2021-12-30 13:41:43
Created container mysql	Created	cloudmonitor	 正常 	vm-50-test	Pod	mysql-mm-sts-0	2021-12-30 13:41:43
Successfully pulled image "10.1.11.20	Pulled	cloudmonitor	• 正常	vm-50-test	Pod	mysql-mm-sts-0	2021-12-30 13:41:42
Pulling image "10.1.11.205/k8s-deploy	Pulling	cloudmonitor	 正常 	vm-50-test	Pod	mysql-mm-sts-0	2021-12-30 13:41:21
Started container k8skafka	Started	cloudmonitor	• 正常	vm-50-test	Pod	kafka-0	2021-12-30 13:41:20
Started container init-mysql	Started	cloudmonitor	 正常 	vm-50-test	Pod	mysql-mm-sts-0	2021-12-30 13:41:20
Successfully pulled image "10.1.11.20	Pulled	cloudmonitor	 正常 	vm-50-test	Pod	kafka-0	2021-12-30 13:41:19
Created container k8skafka	Created	cloudmonitor	 正常 	vm-50-test	Pod	kafka-0	2021-12-30 13:41:19
Created container init-mysql	Created	cloudmonitor	 正常 	vm-50-test	Pod	mysql-mm-sts-0	2021-12-30 13:41:19
Successfully pulled image "10.1.11.20	Pulled	cloudmonitor	 正常 	vm-50-test	Pod	mysql-mm-sts-0	2021-12-30 13:41:19
				第 1-	-10 条/总共 20572	条 < 1 2 ;	3 4 5 … 2058 > 10条/页 <

3.8.3. 日志检索

对于平台支撑组件的日志以及运行在平台之上的算法应用的日志,平台应当提供统一的接口让用户可以进行过滤和查询:

- 日志可选定具体的日志文件和时间段进行查询
- 平台界面可以设置日志的来源进行检索,支持日志文件与标准输出
- 提供关键字模糊匹配的日志查询接口,页面搜索通过API接口请求数据,服务接口进行日志的搜索查询,关键字匹配出可高亮
- 提供日志导出接口,分为时间段导出和搜索结果导出

3.8.3.1. 日志文件



·(III) ,	LILLIL	KO3 A	M+. KO-00-06V		L1007-100-	LINXIT	
* 工作负载:	apm-es-server	× P(DD: apm-es-server-0		* 容器名称:	apm-es-server	
关键字:	error	显示行	数: 1000		* 日志路径:	logs	
*日志文件:	apm-es-server-warn.log				重置	(询 导出	
Bitti -							
Liava Jana Th							
gavallang. In	read.run(1nread.java:/48) [?:1.8.0_2/5]	-0					
RROR] 2021	read.run(1nread.java:/48) [?:1.8.0_2/5] -12-30 13:48:40,286 Aerospike:75 - erro	or happens when inserting into yanshi_yar	nshi_default by key:INDICATOR_ALL_	_DEVICE_STATE with value:{}			
RROR] 2021 va.lang.NullP	read.run(1nread.java:/48) [?:1.8.0_2/5] I-12-30 13:48:40,286 Aerospike:75 - erro PointerException: null	or happens when inserting into yanshi_yar	nshi_default by key:INDICATOR_ALL_	_DEVICE_STATE with value:{}			
RROR] 2021 va.lang.NullP torg.vlis.apm	read.run(1nread.java:/48) [21.8.0_2/5] I-12-30 13:48:40,286 Aerospike:75 - erro PointerException: null n.server.search.cache.Aerospike.insert(Aer	w happens when inserting into yanshi_yar	nshi_default by key:INDICATOR_ALL_	_DEVICE_STATE with value:{}			
ERROR] 2021 va.lang.NullP torg.vlis.apm torg.vlis.apm	read.run(Inread.java:/49)[:1.8.0_2/5] -12-30 13:48:40,286 Aerospike:r56 - erro PointerException: null n.server.search.cache.SearchCache.in(Sea n.server.search.cache.SearchCache.in(Sea Denver.search.cache.SearchCache.in(Sea	or happens when inserting into yanshi_yar ospike.java:72) [apm-es-server-0.0.1-Sh rchCache.java:18] [apm-es-server-0.0.1- ScaretTool: indicates/IV.orgiasState(Mari	nshi_default by key:INDICATOR_ALL IAPSHOT.jar:UNNAMED.615.974cd9 -SNAPSHOT.jar:UNNAMED.615.974d	_DEVICE_STATE with value:{} bb] cd9b] 		051	
EROR] 2021 va.lang.NullP torg.vlis.apm torg.vlis.apm org.vlis.apm	read.run(11rread.java?r4b) (?:1.8.0_27b) 1-12-30 13:48:40,286 Aerospike:75 - arro olnterException: null n.server.search.cache.Aerospike.insert(Aer n.server.search.cache.SearchCache.in(Sea n.server.search.cache.task.MonitorScreent nonitorScreent Newing Carbon Stache.SearchCache.in(Sea	¹ happens when inserting into yanshi_yar ospike.java:72) [apm-es-server-0.0.1-Sh rchCache.java:18] (apm-es-server-0.0.1- SearchTask.indicatorAlDeviceState(Monit SearchTask.ing/UApriceScate(Monit)	nshi_default by key:INDICATOR_ALL IAPSHOT.jar:UNNAMED.615.974cd9 -SNAPSHOT.jar:UNNAMED.615.974c orScreensBearchTask.java;254) [apm- i-wr:21] [com-ge-spector 0.1 _ SNA	_DEVICE_STATE with value:{} bb] cd9b] es-server-0.0.1-SNAPSHOT.jp PSHOT incl UNAMED 615.07.	ar:UNNAMED.615.974cd	96]	
torg.vlis.apm corg.vlis.apm corg.vlis.apm corg.vlis.apm corg.vlis.apm corg.vlis.apm	read.run() Inread.java:r49 [(?: 1.8.0.279) 1-12-30 [3:48:40,286 Aerospike:75 - enro PointerException: null .server.search.cache.Aerospike.Insert(Aer .server.search.cache.task.MonitorScreen .server.search.cache.task.MonitorScreen .server.search.cache task.MonitorScreen	w happens when inserting into yanshi_yar ospike,java:72) [apm-es-server-0.0.1-SN chCache,java:8] [apm-es-server-0.0.5] SearchTask.indicatorAlDeviceState(Monit SearchTask.run(MonitorScreenSearchTask Sararbar num(MonitorScreenSearchTask	nshi_default by key:INDICATOR_ALL IAPSHOT.jar:UNNAMED.615.974cd9 SNAPSHOT.jar:UNNAMED.615.974cd orScreenSearchTask.java:254) [apm- java:71) [apm-es-server-0.0.1-SNA 20] [apm-es-server-0.0.1-SNAPSH	_DEVICE_STATE with value:{} bb] cd9b] -es-server-0.0.1-SNAPSHOT.jr PSHOT.jar.UNNAMED.615.974.cf91	ar:UNNAMED.615.974cd: 4cd9b] ol	96)	
torg.vlis.apm torg.vlis.apm torg.vlis.apm torg.vlis.apm torg.vlis.apm torg.vlis.apm torg.vlis.apm	read.rum(Inread.javz.r49)[r1.8.0_2/8) r1.2-30 13.48(A).286 Aerospike:75 - time rointerException: null a.server.search.cache.Aerospike.insert(Aer I.server.search.cache Asak.MonitorSoreent server.search.cache task.MonitorSoreent s.server.search.cache task.MonitorSoreent s.server.search.cache.task.MonitorSoreent s.server.search.cache.task.MonitorSoreent s.server.search.cache.task.MonitorSoreent	whappens when inserting no sanshi_yar ospike.java:72) [apm-es-server-0.0.1-Sh rchCache.java:18) [apm-es-server-0.0.1- searchTask.indicatorAlloeviceState(Monti Searchtark.indicatorAlloeviceState(Monti S	nshi_default by key:INDICATOR_ALL IAPSHOT,jar:UNNAMED.615.97408 SNAPSHOT.jar:UNNAMED.615.974 GriceenSearchTask.java.254) [apm :java:71) [apm-es-server-0.0.1-SNA 29) [apm-es-server-0.0.1-SNAPSH/	_DEVICE_STATE with value:{} bb] cd9b] es-server-0.0.1-SNAPSHOT.j VPSHOT.jar:UNNAMED.615.974cd9J OT.jar:UNNAMED.615.974cd9J	ar:UNNAMED.615.974cd 4cd9b] 5]	96)	
Fava.lang. In ERROR] 2021 iva.lang.NullP t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm java.util.con	read.rum(Inread.javz.rka) [r1.8.0_r/s) r1.2-30 13.48(40.286 Aerospike.75 - Brrc volaterException: null I.server.search.cache.Aerospike.insert(Aer I.server.search.cache.SaarchCache in(Sea I.server.search.cache.task.MonitorScreent I.server.search.task.MonitorScreent I.server.search.task.MonitorScreent I.server.search.task.MonitorScreent I.server.search.task.MonitorScreent I.server.search.task.MonitorScreent I.server.search.task.MonitorScreent I.server.search.task.MonitorScreent I.server.search.task.MonitorScreent I.server.s	spike.java:72) [apm-es-server-0.0.1-SM ospike.java:72) [apm-es-server-0.0.1-SM ortiCache.java:18] [apm-es-server-0.0.1- SearchTask.indicatorAllDeviceState(Monitis SearchTask.indicatorAllDeviceState(Monitis SearchTask.indicatorAllDeviceState(Monitis Searchar:JundMonitorScreenSearcharjava: Secutors.java:511) [7:1.8.0.275] sci Java:781 [2:1.8.0.275]	nshi_default by key:INDICATOR_ALL_ IAPSHOT.jar:UNNAMED.615.974cd9 SNAPSHOT.jar:UNNAMED.615.974 orScreenSearchTask.java:254) [apm- ciava:71) [apm-es-server-0.0.1-SNA 229] [apm-es-server-0.0.1-SNAPSHI	_DEVICE_STATE with value:{} bb] cd9b) -es-server-0.0.1-SNAPSHOT.ji VSHOT.jar:UNNAMED.615.974cd9) OT.jar:UNNAMED.615.974cd9)	ar:UNNAMED.615.974cd 4cd9b] p]	95]	
rava.tang.1n ERROR] 2021 tva.lang.NullP t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm java.util.con java.util.con	read-run() mread-javz.rka) [r1.8.0_r/8) r1.2-s0 13:48(A).286 Arcspike:75 - tim/ biointerException: null server.search.cache.Arcspike.insert(Aer n.server.search.cache.AsskonfortorScreenf server.search.cache task.MonitorScreenf server.search.cache task.MonitorScreenf server.search.cache task.MonitorScreenf server.search.cache task.MonitorScreenf current.ExecutorSRunnableAdapter.cali(f current.EvecutorSRunnableAdapter.calif(server.search.earb.task.MonitorScreenf server.search.earb.task.MonitorScreenf server.search.earb.task.MonitorScreenf server.search.earb.task.MonitorScreenf server.search.earb.task.MonitorScreenf server.search.earb.task.MonitorScreenf server.search.earb.task.MonitorScreenf server.search.earb.task.task.task.task.task.task.task.task	Appens when inserting into vashi_var sospike.java:72) [apm-es-server-0.0.1-S rchCache.java:18) [apm-es-server-0.0.1-S rchCache.java:18) [apm-es-server-0.0.1-S searchTask.und/notiorScreensearchTask searchar.run(MonitorScreensearchTask searchar.run(MonitorScreensearchTask securots:java:211) [7:1.8.0_276] sk.java:308) [7:1.8.0_275] sk.java:308) [7:1.8.0_275]	nshi_default by key:INDICATOR_ALL IAPSHOT.jar:UNNAMED.615.974cd8 SNAPSHOT.jar:UNNAMED.615.974 orScreenSearchTask.java:254) [apm- java:71) [apm-es-server-0.0.1-SNA 29) [apm-es-server-0.0.1-SNAPSHi lenThrcadDoolSverutor Java:180) [2]	_DEVICE_STATE with value:{} bb] cd9b] es-server-0.0.1-SNAPSHOT.ja PSHOT.jar.UNNAMED.615.97 oT.jar.UNNAMED.615.974cd9/ 1.8.0.275]	17:UNNAMED.615.974cd 4cd9b] o]	96)	
rava.tarig.10 ERROR] 2021 voa.lang.NullP t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t java.util.con- java.util.con- i java.util.con-	read.rum() mead.javz.rka) [r1.8.0_z/r3) r1.2-30 13.48(A).286 Aerospike.75 - gind rointerException: null server.search.cache.Aerospike.insert(Aer server.search.cache.SearchCache.in(Sea server.search.cache.task.MonitorScreent server.search.cache.task.MonitorScreent server.search.cache.task.MonitorScreent current.ExecutorsRnumableAdapter.cali() current.ExecutorsRnumableAdapter.cali() current.ScheduledThreadDoolSxecutorSS	whappens when inserting into vashi_var ospike.java:72) [apm-es-server-0.0.1-Sh crbCache.java:18) [apm-es-server-0.0.1- SearchTask.indicatorAlloeviceState(Monitio SearchTask.indicatorAlloeviceState(nshi_default by key:INDICATOR_ALL, IAPSHOT.jar:UNNAMED.615.974cdg SNAPSHOT.jar.UNNAMED.615.974 ofScreenSearchTask.java:25d1 japm .java:71 japm-es-srever-0.0.1-SNA 29) [apm-es-server-0.0.1-SNAPSHI ledThreadPoolExecutor java:180) [?: 180.015/secutor java:240.12:18.0.27	_DEVICE_STATE with value:{} pb] cdb] es-server-0.0.1-SNAPSHOT;i psHoTjar:UNNAMED.615.974cd9) 1.8.0_275] 51	ir:UNNAMED.615.974cd 4cd9b] 5]	96)	
rava.tarig.1n ERROR] 2021 va.lang.NullP t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t java.util.con- java.util.con- java.util.con-	read.rum(Inread.javz.rka) [r1.8.0_r/s) 1-2-30 13:48(40,286 Aerospike.75 - Brrc volaterException: null server.search.cache.Aerospike.insert(Aer server.search.cache.task.MonitorSoreen server.search.cache.task.MonitorSoreen server.search.cache.task.MonitorSoreen server.search.cache.task.MonitorSoreen server.search.cache.task.MonitorSoreen server.search.cache.task.MonitorSoreen current.EvecutorsRunnableAdapter.call(U current.FutureTask.runAndReset(FutureTa current.ScheduledThreadPoolExecutorSB current.ScheduledThreadPoolExecutorSB	spike.java:72) [apm-es-server-0.0.1-SM ospike.java:72) [apm-es-server-0.0.1-SM ortCache.java:81 [apm-es-server-0.0.1- SearchTask.indicatorAllDeviceState(Monito SearchTask.indicatorAllDeviceState(Monito Searchart.java:5011) [?:1.8.0_276] sk.java:308) [?:1.8.0_275] cheduledFutureTask.accesS301[ScheduledThreas vaeIchorLiver.java:15012;1.8.0_275]	nshi_default by key:INDICATOR_ALL, IAPSHOT.jar:UNNAMED.615.974cdg SNAPSHOT.jar.UNNAMED.615.974 orScreenSearchTask.java:254) [apm -java:71) [apm-es-server-0.0.1-SNA 29] [apm-es-server-0.0.1-SNAPSH [adThreadPcolExecutor.java:180) [7: dPcolExecutor.java:294) [7:1.8.0_27 *5	DEVICE_STATE with value:{} bb] cdb] -es-server-0.0.1-SNAPSHOT.ji PSHOTJar:UNNAMED.615.97 oT.jar:UNNAMED.615.974cd9J 1.8.0_275] 5]	n:UNNAMED.615.974cd (cd9b) o]	96]	
rava.tarig.101 ERROR] 2021 t org.vlis.apm t org.vli	read.rum(Inread.javz.rab)[r1.8.0_z/r3) solmterException: null solmterException: null sourcestate: Cache Aerospike.insert(Aer server.search.cache Assk.MonitorScreent server.search.cache task.MonitorScreent server.search.cache task.MonitorScreent server.search.cache task.MonitorScreent server.search.cache task.MonitorScreent current.ExecutorSRumableAdapter.cali(furure1 acurrent.ExecutorSRumableAdapter.calif current.ExecutorSRumableAdapter.calif current.ExecutorSRumableAdapter.calif current.ExecutorSRumableAdapter.calif current.ExecutorSRumableAdapter.calif current.ExecutorSRumableAdapter.calif current.ThreadPoolExecutorSB	Appens when inserting into yanshi_yar cosplike.java:72) [apm-es-server-0.0.1-S rchCache.java:18) [apm-es-server-0.0.1-S searchTask.indicatorXilDeviceState(Monit SearchTask.indicatorXilDeviceState(Monit Security: Java:19) [7:1.8.0_275] Sk.java:308) [7:1.8.0_275] cheduledFuturTask.run(ScheduledThreas readPoolSxecutor.java:149) [7:1.8.0_27	nshi_default by key:INDICATOR_ALL, LAPSHOT.jar:UNINAMED.615.974cd9 SNAPSHOT.jar:UNINAMED.615.974 of ScreenSearchTask.java:2641 [apm- java:71] [apm-es-server-0.0.1-SNA 29) [apm-es-server-0.0.1-SNAPSHI ledThreadPoolExecutor.java:180) [7: dPoolExecutor.java:294) [7:1.8.0_27 (5]	_DEVICE_STATE with value:{} bb] cd9b] cd9b] es-server-0.0.1-SNAPSHOT,ji pSHOT,jar:UNNAMED.615.974cd9j 1.8.0_275] 5]	ir:UNNAMED.615.974cd (cddb) o]	96)	
rava.lang.1n ERROR] 2021 t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t java.util.con java.util.con java.util.con java.util.con	read.rum() mread.javz.r49 [r1.8.0_z/r9] r1.2-30 13.48(40.286 Aerospike.76 - grrd rohinterScoption: null server.search.cache.Aerospike.insert(Aer server.search.cache.SaerchCache.in[Sea server.search.cache.SaerchCache.in[Sea server.search.cache.task.MonitorScreen server.search.cache.task.MonitorScreen server.search.cache.task.MonitorScreen current.Euterafsk.runAndReset(FuturEra current.ScheduledThreadPoolExecutor\$S current.ScheduledThreadPoolExecutor\$S current.ScheduledThreadPoolExecutor\$S current.ScheduledThreadPoolExecutor\$S current.ThreadPoolExecutor\$Worker(TI) current.ThreadPoolExecutor\$Worker(TI) current.ThreadPoolExecutor\$Norker(TI) current.ThreadPoolExecutor\$Norker(TI) R 0, 2751) R 0, 2751 R	spike.java:72) [apm-es-server-0.0.1-SM tofCache.java:72) [apm-es-server-0.0.1-SM tofCache.java:18] [apm-es-server-0.0.1- SearchTask.indicatorAllDeviceState(Mohit SearchTask.indicatorAllDeviceState(Mohit SearchTark.run(MohitorScreensSearchTask Searcher.run(MohitorScreensSearchTask skjava:308] [:18.0.275] cheduledFutureTask.access\$301(ScheduledThreadPolieKeutor.java:149) [:1.8.0.275] theadPolieKeutor.java:149] [:1.8.0.275] threadPoolExecutor.java:169] [:1.8.0.275]	nshi, default by key:INDICATOR_ALL, NAPSHOT.jar:UNNAMED.615.974cd9 SANPSHOT.jar.UNNAMED.615.974 orScreenSearchTask.java:254) [ppm- .java:71) [apm-es-server-0.0.1-SNA 229] [apm-es-server-0.0.1-SNAPSHI ledThreadPoolExecutor.java:180) [?: 4PoolExecutor.java:284) [?:1.8.0_27 6]	_DEVICE_STATE with value:{} bb] cd9b] =s=server-0.0.1-SNAPSHOT.jr PSHOT.jar;UNNAMED.615.974 dTjar;UNNAMED.615.974cd9l 1.8.0_275] 5]	nr:UNNAMED.615.974cd (cd9b) o)	96)	
rava.tarig.1n RROR] 2021 Iva.lang.NullF t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t java.util.con- t java.util.con- t java.util.con- java.util.con- java.util.con- java.atil.con- java.atil.con-	read.rum(Inread.javz.rka) [r1.8.0_2/8] r1.2-30 13:48(A) 286 Arcspike:r5.5 tim obinterException: null server.search.cache.Xenspike.insert(Aer server.search.cache Xask MonitorSoreent server.search.cache task MonitorSoreent server.search.cache task MonitorSoreent server.search.cache task MonitorSoreent current ExecutorSRunnableAdapter.cali(K current ExecutorSRunnableAdapter.cali(K current.ScheduledThreadPoolExecutorS current.ScheduledThreadPoolExecutorS current.ThreadPoolExecutorSRunnableAdapter.cali(K current.ThreadPoolExecutorSectorS current.ScheduledThreadPoolExecutorS Securent.ScheduledThreadPoolExecutorS (1.3.2.0) 13:48(1.2.1.8.0.275)	Appens when inserting into vashi_var cospike.java:72) [apm-es-server-0.0.1-S rchCache.java:18) [apm-es-server-0.0.1- SearchTask.und/notroscreensearchTask searcher.run(MonitorScreensearchTask searcher.run(MonitorScreensearchTask sexuctors.java:51) [?:1.8.0_276] sk.java:308) [?:1.8.0_276] cheduledFutureTask.un(ScheduledThreat readPoolExecutor.java:624) [?:1.8.0_277 ThreadPoolExecutor.java:624) [?:1.8.0_277	nshi_default by key:INDICATOR_ALL_ IAPSHOT.jar:UNINAMED.615.974cd SNAPSHOT.jar:UNINAMED.615.974cd orScreenSearchTask.java:2641 japm: java:711 japm-es-server-0.0.1-SNA 29) japm-es-server-0.0.1-SNAPSH ledThreadPoolExecutor.java:1800 [?: PPoolExecutor.java:294) [?:1.8.0_27 f5] f5] pbil_default_by.key.JONITOR_SCREEN	DEVICE_STATE with value:{} bb] cdb] -es-server-0.0.1-SNAPSHOT.ja vGbDT.jar:UNNAMED.615.97 oT.jar:UNNAMED.615.974cd9J 1.8.0_275] 5] EN_SMAIL_HOST with value/	11:UNNAMED.615.974cd (cd9b) c]	96) 1971 - Success ^a Brin, ^b rantadan	
rava tang, In RROR] 2021 va.lang.NullF t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm t org.vlis.apm j ava.util.con j ava.util.con j ava.util.con j ava.util.con j ava.util.con j ava.util.con j ava.util.con j ava.util.con	read.rum(Inread.pavz.498) [r1.8.0_z/9] r1.2-30 13:48:40,286 Aerospike:75 - girrd sharerx:search.cache.Aerospike.insert(Aer server.search.cache.Search.cache.in(Sea server.search.cache.Search.cache.in(Sea server.search.cache.task.MonitorScreent server.search.cache.task.MonitorScreent server.search.cache.task.MonitorScreent current.Eurecutors\$RunnableAdapter.call(current.EureSkrunnARGeek(FutureTa current.Earbackunstangeek(FutureTa current.ScheduledThreadPoolExecutor\$S current.ThreadPoolExecutor\$Worker.Un(current.ThreadPoolExecutor\$Worker.Un(server.scheduledThreadPoolExecutor\$S current.ThreadPoolExecutor\$Worker.Un(server.ScheduledThreadPoolExecutor\$S current.ThreadPoolExecutor\$Worker.Un(server.ScheduledThreadPoolExecutor\$S server.ScheduledThreadPoolExecutor\$S server.ScheduledThreadPoolExecutor\$Worker.Un(server.ScheduledThreadPoolExecutor\$Worker.Un(server.ScheduledThreadPoolExecutor\$S server.ScheduletThreadPoolExecutor\$S server.ScheduletThreadPoolExecutor\$Worker.Un(server.ScheduletThreadPoolExecutor\$Worker.Un(server.ScheduletThreadPoolExecutor\$Worker.Un(server.ScheduletThreadPoolExecutor\$Worker.Un(server.ScheduletThreadPoolExecutor\$Worker.Un(server.ScheduletThreadPoolExecutor\$Worker.Un(server.ScheduletThreadPoolExecutor\$Worker.Un(server.ScheduletThreadPoolExecutor\$Worker.Un(server.ScheduletThreadPoolExecutor\$Worker.Un(server.ScheduletThreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecutor\$Worker.ScheduletStreadPoolExecut	source of the setting into yanshi_yar ospike.java:72) [apm-es-server-0.0.1-Sh crchCache.java:18) [apm-es-server-0.0.1-Sh crchCache.java:18] [apm-es-server-0.0.1- SearchTask.ind/notiorScreenSearchTask Searchars.run(MonitorScreenSearchTask Searchars.run(MonitorScreenSearchTask Searchars.run(MonitorScreenSearchTask Searchars.run(MonitorScreenSearchTask Searchars.run(MonitorScreenSearchTask Searchars.run(MonitorScreenSearchTask Searchars.run(ScheduledThrea readPoolExecutor.java:1149) [?:1.8.0_273 ThreadPoolExecutor.java:624) [?:1.8.0_273 Searchars.win(ScheduledThreatscreenSearchars)]	nshi_default by key:INDICATOR_ALL, IAPSHOT.jar:UNNAMED.615.974cd9 SNAPSHOT.jar.UNNAMED.615.974 orScreenSearchTask.java:2561 [apm- java:71] [apm-es-server-0.0.1-SNA 29] [apm-es-server-0.0.1-SNAPSHI [defThreadPoolExecutor.java:180] [?: dPoolExecutor.java:294] [?:1.8.0_27 [5] [5] Ishi_default by key:MONITOR_SCREE	_DEVICE_STATE with value:{} pb] cdb] es-server-0.0.1-SNAPSHOT;i psHoTjar:UNNAMED.615.97 to jar:UNNAMED.615.974cd9 1.8.0_275] 5] EN_SMALL_HOST with value:{	Ir:UNNAMED.615.974cd ted9b] o] "entity":{"code":200,"dat	9b] ta*:[],"success":true),"metadata"	:(),*status*:200}
ERROR] 2021 ERROR] 2021 ERROR] 2021 torg.vlis.apm torg.vlis.ap	read-truit (nread-javz-/ab) [r1.8.0_2/79) lointerException: null server.saarch.cache.Aerosylike.insert(Aer server.saarch.cache.Aerosylike.insert(Aer server.saarch.cache task.MonitorScreent server.saarch.cache task.MonitorScreent server.saarch.cache task.MonitorScreent server.saarch.cache task.MonitorScreent server.saarch.cache task.MonitorScreent current.ExecutorSRunnableAdapter.call(fu current.ExecutorSRunnableAdapter.call(fu current.ExecutorSRunnableAdapter.call(fu current.ScheduledThreadPoolExecutorSS current.InreadPoolExecutorSS current.InreadPoolExecutorSS current.InreadPoolExecutorSS current.InreadPoolExecutorSS locationsContexters(1-12.0.2,275) 1-2.301348.40.299 Aerospike:75 - arr olointerException: null	Appens when inserting into yanshi_yar sopike.java:72) [apm-es-server-0.0.1-S rchCache,java:18] [apm-es-server-0.0.1-S rchCache,java:18] [apm-es-server-0.0.1-S searchTask.indicatorAllDeviceState(Monit searchTask.undKontorScreenSearchTask searcher.run(MonitorScreenSearchTask searcher.run(MonitorScreenSearchEarcher.run(MonitorScreenSearcher.run(ScheduledThreat) *********************************	shi, default by key:INDICATOR_ALL, APSHOT.jar:UNNAMED.615.974cd9 SANPSHOT.jar.UNNAMED.615.974 orscreensearchTask.java:254) [apm- sjava:71) [apm-es-server-0.0.1-SNA 29] [apm-es-server-0.0.1-SNAPSHI ledThreadPoolExecutor.java:180) [?: dPoolExecutor.java:284) [?:1.8.0_27 f5] shi_default by key:MONITOR_SCRE MONELOT.ion!INNAMED.615.074.df	DEVICE_STATE with value:{ b] cdb] -es-server-0.0.1-SNAPSHOT.ji vPSHOT.jar.UNNAMED.615.97 coT.jar.UNNAMED.615.974cd9j 1.8.0_275] 6] EN_SMALL_HOST with value:{	ir:UNNAMED.615.974cd 4cd9b] o] "entity":{"code":200,"dat	9b] ta":[],"success":true),"metadata"	:(),*status*:200}
revarianty, in revealing, in revealing, in revealing, in revealing, and to regulis, appent to regulis, appen	read-Turii (Tread Jav2: A49) [r1:8.0_2/79) spreve: Saerch.cache.Aerospike: 75- time spreve: Saerch.cache Aerospike: Insert (Aer 1. server: saerch.cache Assi Monitor Soreent 5. server: saerch.cache task. Monitor Soreent 5. server: Saerch.del: Monitor Soreent 5. server: Saerch.del: Monitor Soreent 5. server: Saerch.cache Asserver: Monitor Soreent 5. server: Saerch.cache.asserver: Monitor Soreent 5. server: Saerch.cache.Asserver: Saerch.cache 5. server: Saerch.cache.Asrospike: Insert(Aer 5. server: Saerch.cache.Asrospike: Insert(Aer 5. server: Saerch.cache.Asrospike: Insert(Aer 5. server: Saerch.cache.SaerSore.co.ba. Inford 5. server: Saerch.cache.SaerSore.co.ba. Inford 5. server: Saerch.cache.SaerSore.co.ba. Inford 5. server: Saerch.cache.Asrospike: Insert(Aer 5. server: Saerch.cache.Asrospike:	Appens when inserting into yanshi_yar ospike.java:72) [apm-es-server-0.0.1-S rchCache.java:18) [apm-es-server-0.0.1- SearchTask.indicatorAlloeviceState(Monit SearchTask.indicatorAlloeviceState(Monit SearchTask.indicatorAlloeviceState(Monit SearchTask.indicatorAlloeviceState(Monit SearchTask.indicatorAlloeviceState(Monit SearchTask.indicatorAlloeviceState(Monit SearchTask.indicatorAlloeviceState(Monit SearchTask.indicatorAlloeviceState(Monit Calded) [7:1.8.0_275] skipar:208 [7:1.8.0_275] skipar:208 [7:1.8.0_277] threadPoolExecutor.java:624) [7:1.8.0_27 threadPoolExecutor.java:624) [7:1.8.0_27 threadPoolExecutor.java:624] [7:1.8.0_27 threadPoolEx	nshi_default by key:INDICATOR_ALL, IAPSHOT.jar:UNNAMED.615.974cdg SNAPSHOT.jar:UNNAMED.615.974cdg SNAPSHOT.jar:UNNAMED.615.974cdg Jayax:71 japm-es-server-0.0.1-SNAPSHI ledThreadPoolExecutor.java:180) [?: dPoolExecutor.java:294) [?:1.8.0_27 [5] nshi_default by key:MONITOR_SCRE IAPSHOT.jar:UNNAMED.615.974cdg	DEVICE_STATE with values() bb] cdbb] es-server-0.0.1-SNAPSHOT,ji psHoTjar:UNNAMED.615.974cd9) 1.8.0_275] 5] EN_SMALL_HOST with values(bb]	ir:UNNAMED.615.974cd tod9b] o] *entity*:{"code":200,"dat	9b) ta":[],"success":true),"metadata"	:{),*status*:200)
revoluting. In rerRoR1 2022 FERROR1 2022 to org.vilis.apm to org.vilis.apm to org.vilis.apm to org.vilis.apm to org.vilis.apm to ava.util.con tjava.util.con	read.rum(Inread.pavz.r49) [r1.8.0_z/r9] h12-30 13:48:40,286 Aerospike.insert(Aerospike.insert(Aerospike.insert(Aerospike.insert(Aerospike.insert(Aerospike.insert(Aerospike.insert). server.search.cache.Aerospike.insert(Aerospike.inserter.search.cache.task.MonitorScreent server.search.cache.task.MonitorScreent server.search.cache.task.MonitorScreent current.EurotarSa.runnAndRead(FutureTa current.ScheduledThreadPoolExecutorSS current.ScheduledThreadPoolExecutorSS current.ThreadPoolExecutorSWorker.rum(current.InreadPoolExecutorSWorker.rum(server.search.cache.Aerospike.insert(Aerospike.insert(Aerospike.insert(Aerospike.insert(Aerospike.insert)) h12-30 13:48:40,299 Aerospike:75 - arro olinterException: null server.search.cache.Aerospike.insert(Aerospike.insert(Aerospike.insert))	Appens when inserting into yanshi_yar ospike.java:72) [apm-es-server-0.0.1-Sh fclCache.java:18] [apm-es-server-0.0.1-Sh searchTask.indicatorAlloeviceState(Monitio SearchTask.indicatorAlloeviceState(Monitio SearchTask.indicatorAlloeviceState(Monitio SearchTask.indicatorAlloeviceState(Monitio SearchTask.indicatorAlloeviceState(Monitio SearchTask.indicatorAlloeviceState(Monitio cheduledFutureTask.access\$301(Schedu cheduledFutureTask.access\$301(Schedu cheduledFutureTask.access\$301(Schedu cheduledFutureTask.access\$301(Schedu CheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.access\$301(ScheduledFutureTask.	shi, default by key:INDICATOR_ALL, IAPSHOT.jar:UNNAMED.615.974cd9 SNAPSHOT.jar:UNNAMED.615.974 orScreenSearchTask.java:2541 [apm- sava:71) [apm-es-server-0.0.1-SNA 229] [apm-es-server-0.0.1-SNAPSHI ledThreadPoolExecutor.java:180) [?: dPoolExecutor.java:294] [?:1.8.0_27 6] nshi_default by key:MONITOR_SCREI IAPSHOT.jar:UNNAMED.615.974cd8 SNAPSHOT.jar:UNNAMED.615.974cd8	LDEVICE_STATE with value:{} bb] cdb] res-server-0.0.1-SNAPSHOT.jr rPSHOT.jar:UNNAMED.615.974 OT.jar:UNNAMED.615.974cd9/ 1.8.0_275] 5] EN_SMALL_HOST with value:{ bb] cdb] cdb]	Ir:UNNAMED.615.974cd (acd9b) o) "entity":{"code":200,"dat	9b] ta":[],"success":true),"metadata"	:{),"status":200}
revarianty, in representation, in representation, in representation of the representatio	read-truit (nread-javz.rab) [r1.8.0_z/r3) solver.55. gint (A) 266 Arospike: T5. gint obinterException: null server.search.cache.SearchCache.in(Sea server.search.cache.SearchCache.in(Sea server.search.cache.task.MonitorSoreent server.search.cache.task.MonitorSoreent server.search.cache.task.MonitorSoreent server.search.cache.task.MonitorSoreent current.ExecutorsRumableAdapter.calif(current.ExecutorsRumableAdapter.calif(current.ExecutorsRumableAdapter.calif(current.ExecutorsRumableAdapter.calif(current.ExecutorsRumableAdapter.calif(current.ExecutorsRumableAdapter.calif(current.ExecutorsRumableAdapter.calif(current.Executor.rumWorker(T1 read.rum(Thread.pivz.r348) [r1.8.0_275] 1-12-30 13:48:40,299 Aerospike:75 - gint server.search.cache.SearchCache.In(Sea server.search.cache.SearchCache.In(Sea server.search.cache.SearchCache.Search.In(Sea server.search.cache.SearchCache.Search.In(Sea server.search.cache.SearchCache.Search.In(Sea server.search.cache.SearchCache.Search.In(Sea server.search.cache.Search.Search.In(Sea server.search.cache.Search.Search.In(Sea server.search.cache.Search.In(Sea server.search.In(Search.In(Sea ser	Appens when inserting into yanshi_yar cospike.java:72) [apm-es-server-0.0.1-Sh rchCache.java:18) [apm-es-server-0.0.1-Sh rchCache.java:18) [apm-es-server-0.0.1- SearchTask.und/notroScreensearchTask searcher.run(MonitorScreensearchTask searcher.run(MonitorScreensearchTask secutors.java:11) [21.18.0.275] sk.java:308) [21.8.0.275] sheduledFuturTask.run(ScheduledThreat readPoolExecutor.java:1149) [21.8.0.27 hreadPoolExecutor.java:624] [21.8.0.27 ja happens when inserting into yanshi_yar cospike.java:72) [apm-es-server-0.0.1- searchTask.hostIps(MonitorScreenSearchTo.1 searchTask.hostIps(MonitorScreenSearchTo.1 searchTask.hostIps(MonitorScreenSearchTo.1)	nshi_default by key:INDICATOR_ALL, LAPSHOT.jar.UNINAMED.615.974cd SNAPSHOT.jar.UNINAMED.615.974cd SNAPSHOT.jar.UNINAMED.615.974cd 2015creenSearchTask.java:264) [apm- java:71) [apm-es-server-0.0.1-SNAPSHI [edThreadPoolExecutor.java:180) [?: dPoolExecutor.java:284) [?:1.8.0_77 [6] nshi_default by key:MONITOR_SCRE ANDSHOT.jar.UNINAMED.615.974cd SNAPSHOT.jar.UNINAMED.615.974cd	DEVICE_STATE with value:{) bb] deb] deb] des-server-0.0.1-SNAPSHOT.j; UPSHOT.jar:UNNAMED.615.974cd9) 1.8.0_275] 6] EN_SMALL_HOST with value:{ bb] deb] deb] deb]	Ir:UNNAMED.615.974cd (cd9b) c] "entity":{"code":200,"dat 5.974cd9b]	9b] ta":[],"success":true),"metadata"	:(),*status*:200)
reventing. In reversion of the second to registic approximation to registic approximation registic approximation registic registic approximation registic approxim	read-Tum(Inread-Jav2: A91 [r1:8.0_2/*9] r1:2-3013:48(A).286 Aerospike: 75 - time interException: null server: search.cache. Aerospike.insert(Aer server: search.cache. SearchCache.in(Sea server: search.cache task. MonitorSoreent server: search.cache task. MonitorSoreent server: search.cache task. MonitorSoreent current.ExecutorsSRunnableAdapter.call(current.ExecutorsSRunnableAdapter.call) current.ExecutorsSRunnableAdapter.call current.ScheduledThreadPoolExecutorSS current.ThreadPoolExecutorSW current.ThreadPoolExecutorSW current.ThreadPoolExecutorSW server: search.cache.Aerospike.insert(Aer server: search.cache.Aerospike.insert(Aer server: search.cache.Aerospike.insert(Aer server: search.cache.Aerospike.insert(Aer server: search.cache.Aarospike.insert(Aer server: search.cache.Aarospike.insert(Aer server: search.cache.Aarospike.insert(Aer server: search.cache.Aarospike.insert(Aer server: search.cache.Aarospike.insert(Aer server: search.cache.Aarospike.insert(Aer server: search.cache.Aarospike.insert(Aer server: search.cache.Aarospike.insert(Aer server: search.cache.task.MonitorSoreent	source of the second se	nshi_default by key:INDICATOR_ALL, NAPSHOT.jar:UNNAMED.615.974cd9 SNAPSHOT.jar.UNNAMED.615.974cd9 SNAPSHOT.jar.UNNAMED.615.974 rofscreenSearchTask.java:25d1 [apm- gava:71] [apm-es-server-0.0.1-SNAPSHI ledThreadPoolExecutor.java:180) [?: dPoolExecutor.java:294) [?:1.8.0_27 [5] rshi_default by key:MONITOR_SCREE IAPSHOT.jar.UNNAMED.615.974cd9 SNAPSHOT.jar.JNNAMED.615.974cd9 SNAPSHOT.jar.JNNAMED.615.974cd9 SNAPSHOT.jar.JNNAMED.615.974cd9 SNAPSHOT.jar.JNNAMED.615.974cd9 SNAPSHOT.jar.JNNAMED.615.974cd9 SNAPSHOT.jar.JNNAMED.615.974cd9 SNAPSHOT.jar.JNNAMED.615.974cd9 SNAPSHOT.jar.JNNAMED.615.974cd9 SNAPSHOT.jar.JNNAMED.615.974cd9 SNAPSHOT.jNNAMED.615.974cd9 SNAPSHOT.jNNAMED.615.974cd9 SNAPSHOT.jNNAMED.615.974cd9 SNAPSHOT.jNNAMED.615.974cd9 SNAPSHOT.jNNAMED.615.974cd9 SNAPSHOT.jNNAMED.615.974cd9 SNAPSHOT.jNNAMED.615.974cd9 SNAPSHOT.jNNAMED.615.974cd9 SNAPSHOT.jNNAMED.615.974c09 SNAPSHOT.jNNAMED.615.974c09 SNAPSHOT.jNNAMED.	LDEVICE_STATE with values() Pb] cdsb] es-server-0.0.1-SNAPSHOT,ij vesHoT,iar:UNNAMED.615.974cd9) 1.8.0_275] 5] EN_SMALL_HOST with values(Pb] cdsb] PSNAPSHOT,iar:UNNAMED.615.97 or icruiteNNAMED.615.97	ir:UNNAMED.615.974cd ted9b] o] "entity":{"code":200,"dat 5.974cd9b] ted9b]	9b] ta*:[],"success":true),"metadata"	:(),*status*:200)
reventing. In reversion of the second second to regulate and the second second second second to regulate and the second second second second to regulate and the second second second second second to regulate and the second second second second second to regulate and the second second second second second second to regulate and the second second second second second second to regulate and the second second second second second second second second to regulate and the second se	read-truit (nread-javz-/a9) [r1.8.0_2/79) lointerException: null server-saarch.cache.Aerospike.insert(Aer server-saarch.cache Asarch/Cache.in (Sea server-saarch.cache task. MonitorScreent server-saarch.cache task. MonitorScreent server-saarch.cache task. MonitorScreent server-saarch.cache task. MonitorScreent current.ExecutorSRunnableAdapter.cali(fucurent current.ScheduledThreadPoolExecutorSS current.ScheduledThreadPoolExecutorSS current.ScheduledThreadPoolExecutorSS current.ThreadPoolExecutorSS current.ThreadPoolExecutorSS current.ThreadPoolExecutorSS current.ThreadPoolExecutorSS server-saarch.cache AsarchCache.in(Sea server-saarch.cache AsarchCache.in(Sea server-saarch.cache.SaarchCache.in(Sea server-saarch.cache.task.MonitorScreent server-saarch.cache.task.MonitorScreent server-saarch.cache.task.MonitorScreent server-saarch.cache.task.MonitorScreent server-saarch.cache.task.MonitorScreent server-saarch.cache.task.MonitorScreent server-saarch.cache.task.MonitorScreent server-saarch.cache.task.MonitorScreent server-saarch.cache.task.MonitorScreent	Mappens when inserting into vanshi_yar ospike.java:72) [apm-es-server-0.0.1-Sh rchCache,java:18) [apm-es-server-0.0.1-Sh rchCache,java:18) [apm-es-server-0.0.1-Sh searchTask.und/notiorScreenSearchTask Searcher.run(MonitorScreenSearchTask Searcher.run(MonitorScreenSearchTask Securotrs.java:51) [7:1.8.0_275] sk.java:308) [7:1.8.0_275] chaduledFutureTask.run(ScheduledThreat readPoolSxecutor.java:624) [7:1.8.0_27 ThreadPoolSxecutor.java:624] [7:1.8.0_27 and papens when inserting into yanshi_yar ospike.java:72) [apm-es-server-0.0.1-Sh fricCache.java:18] [apm-es-server-0.0.1-Sh SearchTask.run(MonitorScreenSearchTask Searcher.run(MonitorScreenSearchTask Searcher.run(MonitorStreenSearchEarchask Searcher.run(MonitorStreenSearchas) Searchask Searcher.run(MonitorStreenSearchas) Searchask Searcher.run(MonitorStreenSearchas) Searchask Searchask Searchask Searchask Searchask Searchask Searchask Searchask Searchask Searchask Searchask Searchask Searchask Searchask Search	nshi_default by key:INDICATOR_ALL, IAPSHOT.jar:UNNAMED.615.974cd SNAPSHOT.jar:UNNAMED.615.974cd SNAPSHOT.jar:UNNAMED.615.974cd Snav:71) (apm-es-server-0.0.1-SNA 29) [apm-es-server-0.0.1-SNAPSHI ledThreadPoolExecutor.java:180) (?: PoolExecutor.java:294) [?:1.8.0_27 f5] shil_default by key:MONITOR_SCREE IAPSHOT.jar:UNNAMED.615.974cd SNAPSHOT.jar.UNNAMED.615.974cd SNAPSHOT.jar.UNNAMED.615.974cd SNAPSHOT.jar.UNNAMED.615.974cd SNAPSHOT.jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.Jar.UNNAMED.615.974cd SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHOT.JAR.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHOT.SNAPSHOT.SNAPSHI SNAPSHI SNAPSHI SNAPSHOT.SNAPSHI SNAPSHI SNA	DEVICE_STATE with value:{} b] b] ses-server-0.0.1-SNAPSHOT.ji vPSHOT.jar:UNNAMED.615.97 407 jar:UNNAMED.615.974cd90 1.8.0_275] 5] EN_SMALL_HOST with value:{ vbj cd9b] -SNAPSHOT.jar:UNNAMED.615.97 4094HOT.jar:UNNAMED.615.974cd90	ar:UNNAMED.615.974cd (acd9b)) *entity*:{"code":200,"dat 5.974cd9b] (acd9b) 2]	9b] ta":[],"success":true),"metadata"	:(),*status*:200)

3.8.3.2. 标准输出

* 租户:	CLOUDMONITOR	✓ * k8s集群:	k8-68-dev	V	日志来源:	标准输出	V
* 工作负载:	mysql-mm-sts	✓ * POD:	mysql-mm-sts-0	> 高级	* 容器名称:	mysql	V
关键字:	not found	显示行数:	1000		历史日志:	西	
时间:	白 7天						
					重置	查询 导出	
日志详情							
enough pe	ermission to load dynamically ge	nerated classes. Please check	the configuration for I	petter performance.		, ou ou ou puin or jou uon i	
[2021-12	-23 08:18:59] main DEBUG io.ne	etty.util.internal.logging.Slf4JL	.ogger.debug(Slf4JLog	ger.java:76) -Dio.netty.nol	PreferDirect: fa	alse	
[2021-12	-23 08:18:59] main DEBUG io.ne	etty.util.internal.logging.Slf4JL	.ogger.debug(Slf4JLog	ger.java:/6) -Dio.netty.nol	KeySetOptimiz	ation: false	
[2021-12	-23 08:18:59] Thread-0 INFO co	om.harmonvcloud.nettv4.daen	non.DaemonRestart.ru	n(DaemonRestart.java:236	i) not found pi	d by process name start apr	plication!
[2021-12	-23 08:18:59] main DEBUG io.ne	etty.util.internal.logging.Slf4JL	.ogger.debug(Slf4JLog	ger.java:76) -Dio.netty.allo	cator.numHea	apArenas: 1	inoution.
[2021-12	-23 08:18:59] main DEBUG io.ne	etty.util.internal.logging.Slf4JL	.ogger.debug(Slf4JLog	ger.java:76) -Dio.netty.allo	cator.numDire	ectArenas: 1	
[2021-12	-23 08:18:59] main DEBUG io.ne	etty.util.internal.logging.Slf4JL	.ogger.debug(Slf4JLog	ger.java:76) -Dio.netty.allo	cator.pageSiz	te: 8192	
[2021-12	-23 08:18:59] main DEBUG io.ne	etty.util.internal.logging.Slf4JL	.ogger.debug(Slf4JLog	ger.java:76) -Dio.netty.allo	ocator.maxOrd	ler: 11	
[2021-12	-23 08:18:59] main DEBUG io.ne	etty.util.internal.logging.Slf4JL	.ogger.debug(Slf4JLog	ger.java:76) -Dio.netty.allo	cator.chunkSi	ze: 16777216	
[2021-12	-23 08:18:59] main DEBUG io.ne	etty.util.internal.logging.Slf4JL	.ogger.debug(Slf4JLog	ger.java:76) Loopback inte	erface: lo		
[2021-12	-23 08:18:59] main DEBUG io.ne	etty.util.internal.logging.Slf4JL	.ogger.debug(Slf4JLog	ger.java:76) Loopback add	ress: /127.0.0	D.1 (primary)	
[2021-12	-23 08:18:59] main DEBUG io.ne	etty.util.internal.logging.Slf4JL	.ogger.debug(Slf4JLog	ger.java:76) /proc/sys/net/	core/somaxco	nn: 128	
[2021-12	-23 08:18:59] main INFO com.ha	armonycloud.netty4.impl.Netty	y4ServiceImpl.start(Ne	etty4ServiceImpl.java:47) S	SERVER starte	d at port 18088 ??????	

3.8.4. 资源检索



资源检索提供对所纳管集群内相关 pod 的查询,查询结果以列表展示。通过对集群、主机、 pod 等条件进行筛选。如下图:

租户: 主机状态:	zzzzzzz 正常		×]	k8s集群: Pod列表:	vm-50-test 全部pod ¥	8	主机列表: Pod状态:	全部	8
时间:	白 5分钟			L					章询 导出
主机		主机运行状态	Pod		PodUID	Pod运行状态	关联业务		更新时间 💠
node52		• 正常	calico-node-cs565		0b45a3d1-de08-4029-90bb	• 正常			2021-12-07 10:54:43
node52		• 正常	front-0		1f8b1951-06e3-404b-bde8-2	• 正常			2021-12-16 14:16:35
node52		• 正常	apm-data-receiver-0		2588285f-498d-41df-a558-a	• 正常	9 <u>000</u> 0		2021-12-16 14:16:35
node52		• 正常	kube-proxy-zgggs		38e74d8f-e05f-4ba2-a3fc-15	• 正常			2021-12-07 10:54:43
node52		• 正常	es-cluster-2		3977cf45-aaf4-4bb8-b171-2f	• 异常②			2021-12-27 13:12:07
node52		• 正常	kafka-2		53dd14ae-a839-49cc-b100-8	• 正常			2021-12-27 13:12:08
node52		• 正常	zk-2		623fc7ea-e5d8-4f51-930e-3	• 正常			2021-12-16 14:16:33
node52		• 正常	apm-es-server-0		b1f04f77-582a-4392-a568-3	• 异常③			2021-12-27 13:12:06
node52		• 正常	hcmine-5ddrj		b5cc5699-9730-4319-9824	• 正常			2021-12-07 10:54:31

3.8.5. 操作审计

操作审计功能是为了记录平台的一些操作历史记录,对回溯故障原因与故障定位有着重要作用。列表如下图:

模块标题	业务类型	操作人员	请求URL	主机地址	请求参数	操作状态		错误消息	操作时间	操作	٢
告警规则	修改	admin	/apmServer-sl/alarm-rul	192.168.175.151	0.0	 正常 	ħ		2021-12-29 10:39:33	详情	
告警规则	修改	admin	/apmServer-sl/alarm-rul	192.168.175.151	0.0	 正常 			2021-12-29 09:03:47	详情	
告警规则	修改	admin	/apmServer-sl/alarm-rul	192.168.175.150	{"mergeInterval":20,"	 正常 			2021-12-28 17:15:48	详情	
告警规则	修改	admin	/apmServer-sl/alarm-rul	192.168.175.150	0.0	 正常 			2021-12-28 17:14:57	详情	
告警规则	修改	admin	/apmServer-sl/alarm-rul	192.168.175.150	0.0	• 正常			2021-12-28 17:14:30	详情	
告警规则	修改	admin	/apmServer-sl/alarm-rul	192.168.175.150	0.0	 正常 			2021-12-28 17:14:28	详情	
告警规则	修改	admin	/apmServer-sl/alarm-rul	192.168.175.150	0.0	 正常 			2021-12-28 17:14:26	详情	
告警规则	修改	admin	/apmServer-sl/alarm-rul	192.168.175.148	0.0	• 正常			2021-12-28 13:46:42	详情	
告警规则	修改	admin	/apmServer-sl/alarm-rul	192.168.175.148	0.0	• 正常			2021-12-28 13:45:57	详情	
告警规则	修改	admin	/apmServer-sl/alarm-rul	192.168.175.148	0.0	 正常 			2021-12-28 13:45:55	详情	

第 1-10 条/总共 631条 < 1 2 3 4 5 *** 64 > 10 条/页 <



3.9. 工具

3.9.1. tcp dump

平台对 tcp dump 工具进行集成,提供了对数据链路层网络数据的抓取能力,对最底层网络数据的分析提供了支持:

* 佳群之称 ·	k8_150_rel
《未好口你。	ко-тор-тег
*命名空间:	hcmine V
POD名称:	
容器名称:	
IPV6:	ON
* 主机IP/子网掩码:	例如: 192.0.2.5/24, 2001:0db8:85a3:8a2a:7334/64
端口:	
协议:	arp tcp udp Icmp

平台对其提供了参数设置的能力,此工具可支持4种抓包协议: arp、tcp、udp、icmp。



3.10. 配置

配置中心提供了对整个系统平台的统一配置页面,根据功能区分为以下9个模块:

3.10.1. 用户管理

此模块提供了对平台用户的新增编辑删除及相关权限配置的能力。

3.10.1.1. 用户列表

账号	名称	邮箱		手机	平台权限	状态	〒 操作	'F	
test	测试	111@13.com		17899999999	普通用户	已启用	编辑	茸 重置密码	删除
test01	测试测试	3213213@qq.com		18248294323	管理员	一日に日用	编辑	単重置密码	删除
test02	测试测试	32132132@qq.com		18248294324	普通用户	已 启用	编	針 重置密码	删除
test03	test	test@qq.com	•	12345678901	普通用户	已 启 用	编	針 重置密码	删除
zanebono	郑丁公大帅逼	qq@qq.com		18808868888	普通用户	已启用	编	車重雷密码	删除
zanebonoalter	郑丁公超级大帅逼	zhengshuaige@niubi.com		18808868889	管理员	一日日日日	编辑	針 重置密码	删除
						第	1-6 条/总共 6	<u>₿</u> < 1) > 10 身

用户列表展示了平台相关用户的信息,在此页面可对用户进行编辑删除与密码相关的修改操 作。

3.10.1.2. 新增用户

点击新增用户按钮,可跳转至新增用户界面,输入相关信息即可新增一个用户。



新建用户配置		Ś
<mark>*</mark> 账号:		
* 名称:		
<mark>*</mark> 密码:		
	密码强度: 弱	
	◎ 只包含大小写字母、数字、符号 (空格除外)	
	◎ 大小写字母、数字、符号包含两种及两种以上组合	
*确认密码:		
*		
* 手机:		
* 邮箱:		
* 部门:		
* 职位:		
平台权限:		\sim
	TTO NIC	74

3.10.2. 租户管理

此模块提供了对平台租户的新增编辑删除及相关权限配置及集群绑定的能力。

3.10.2.1. 租户列表



名称	可用容器云集群	可用主机集群	可见应用	操作 🖑
测试租户1	Ise-68-dev default, ingress-n Ise-59- test-59, copyindex, copyindex, copyindex, copyindex, default, couderon Ise-59- default, monitoring, ingress-n #31 cloudmon bookdemo, monitoring, kube-syst monitoring, cattle-sys #31 kube-syst feet-syst cattle-sys kube-nod kube-nod kube-syst yyzi, kube-publ kube-publ xube-publ	155期 程 共1个	test k8-68-dev 155集群 共2个	in 12 m) 18
CLOUDMONITOR	k8-68-dev cloudmoni 共1个		test4 k8-68-dev 共1个	编辑删除
MONITORING	ks-68-dev homine, cloudmoni yyzi, monitoring,		test4 	编辑 删除
test	k8-68-dev vm-50-te k8-159-rel bookdemo, cloudmoni bookdemo, cloudmoni bookdemo, homine, fleet-syst karte	1955 度将 渐试用盘	test test2 k8-68-dev 155集群 共2个	编辑 删除

租户列表展示了该租户与集群及命名空间的关联关系,赋予租户管理集群的能力。

点击编辑跳转至租户集群命名空间的绑定界面,在此界面可以对租户-集群-命名空间进行 绑定操作。如下图:



	③ 编辑测试租户	1	保存世
	*租户名称:	测试租户1	
	可用容器云集款		
	请选择集群	> 请选择命名空间	添加
可用主机集群	集群	命名空间	操作
	k8-68-dev	defaultingress-nginxcloudmonitorfleet-systemcattle-systemtest-69kube-systemkube-node-leaseyyz1monitoringbookdemokube-publiczanebono	移除
155集群 共1个	k8-59-dev	copyindex test-59 bookdemo	移除
	k8-159-rel	bookdemo default monitoring ingress-nginx fleet-system cattle-system kube-node-lease kube-system kube-public	移除
	可用主机集群 请选择集群	☆ 加	
	集群	类型	操作
	155集群	虚拟机集群	移除
	选择应用		
155集群 测试用虚 共2个	请选择集群名称	◇ 请选择应用 ◇	添加
	应用	集群	操作

3.10.2.2. 新增租户

点击新增租户可跳转至新增租户界面,可进行租户新增与集命名空间绑定。

3.10.2.3. 探针管理

探针管理模块提供了包括主机探针、应用探针的下载功能及已部署探针的查看功能:



彩针	应用探针							
集群: 载探:	全部 ×		节点名称:			۵		重置
1	节点名称	节点IP	所属集群	状态 ② 📑	版本	上次心跳时间 👙	包追踪 ⑦	操作
	10.10.101.72-database-3	10.10.101.72	k8-68-dev	 启用 	v2.0	2021-12-30 14:47:52	() 关闭	休眠 详情
	10.10.101.73-prometheus-data	10.10.101.73	k8-68-dev	• 启用	v2.0	2021-12-30 14:47:52	开启 ●	休眠 详情
		10.10.102.161	k8-68-dev	• 启用	v2.0	2021-12-30 14:47:56	开启	休眠 详情
	10.10.101.68-master	10.10.101.68	k8-68-dev	◎ 休眠	v2.0	2021-12-30 14:47:52	开启	启用 详情
	10.10.101.69-gateway	10.10.101.69	k8-68-dev	• 启用	v2.0	2021-12-30 14:47:52	开启	休眠 详情
		10.10.102.160	k8-68-dev	 启用 	v2.0	2021-12-30 14:47:54	() 关闭	休眠详情
	10.10.102.93-salve	10.10.102.93	k8-68-dev	 启用 	v2.0	2021-12-30 14:47:52	开启	休眠 详情
	10.10.101.70-database-1	10.10.101.70	k8-68-dev	• 启用	v2.0	2021-12-30 14:47:59	开启	休眠 详情
	10.10.101.71-database-2	10.10.101.71	k8-68-dev	• 启用	v2.0	2021-12-30 14:47:59	() 关闭	休眠 详情
	10.10.102.91-salve	10.10.102.91	k8-68-dev	 启用 	v2.0	2021-12-30 14:47:52	开启	休眠 详情

第 1-10 条/总共 11条 < 1 2 > 10 条/页 >

在列表页面可以对已部署探针进行启动、休眠等操作,亦可打开包追踪功能,使探针开启 包追踪的能力。

3.10.3. 全局配置

全局配置模块提供了三个方面的必要配置:组件地址配置、部署地配置、数据中心配置。

3.10.3.1. 组件地址配置



系統配置 ②	
探针路径	/root/config/zip
	设置探针下载时的探针路径。探针模板保存在readme文件夹中。
数据发送地址	apm-data-receiver-svc
	设置数据发送地址,不带端口。不可配置服务名
数据发送端口	57001
	设置数据发送端口。即dc节点6669端口的映射值
配置中心地址	apm-configserver-svc
	私有化模式下,提供所有配置的服务地址
配置中心端口	6659
	通过6659端口获取配置中心配置
自监控心跳检测URL	http://application-monitor/heartbeat
	设置自监控的心跳检测的接口路径。(例:"http://10.10.103.112:9090/application-monitor/heartbeat")
保存配置	

输入组件地址进行各个组件的基础配置。

3.10.3.2. 部署地配置

组件地址配置	部署地配置	数据中心配置				
新增部署地						
部署地				操作		
上海				编辑	删除	
合肥				编辑	删除	
杭州				编辑	删除	

对部署地进行新增、编辑等操作。

3.10.3.3. 数据中心配置



100

组件地址配置	部署地配置	数据中心配置		
新增数据中心				
数据中心配置			部署地	操作
上海数据中心			上海	编辑删除
合肥数据中心			合肥	编辑 删除
杭州数据中心			杭州	编辑删除

对数据中心进行新增、编辑等操作

3.10.4. 集群配置

3.10.4.1. 进程探活

此对进程探活相关参数进行设置,包括进程名、探活方式、端口等。支持两种方式探活: 进程名与端口。

进程探活			
进程名称	探活方式	端口/进程名	
kubelet	进程名	✓ kutjelet	
			取消 保存

3.10.5. 业务配置

此模块对业务相关参数进行设置,包括阈值设置、健康评分、健康检查、url 过滤。

3.10.5.1. 阈值设置

可对业务的慢阈值进行设置



國值设置	健康评分	健康	段检查	URL过滤	
慢事务阈值曹	2置 ②				
慢事务	时间阈值	500	ms		
		设置进行数据	另析的性能阈值	单位为毫秒,缺省值是500毫秒。当接口响应时间超过该阈值时,探针将会记录详细事	务发生数据。
		保存			
		_	۲		

3.10.5.2. 健康评分

可设置健康评分的评分策略,平台根据配置的策略进行业务健康分数的计算。

规则名称搜索	Q										新建健康识
	17.10	响应时间(ms)		错误率(%)		健康检查错误数(次/h) F		Apdex	Pod重启次	'数(次/h)	10.00
观则有标	^{环現} 值区间 权重 值区间 权重	值区间	权重	权重	值区间	权重	操作				
2012-0140 001	K8S	500-2000	10%	0%-40%	10%	1-5	40%	10%	1-5	30%	44+0 721M 8100.
2011年2月20日	虚核	500-2000	20%	0%-40%	20%	1-5	40%	20%			编辑 预览 删陈
	K8S	500-2000	10%	0%-40%	10%	1-5	40%	10%	1-5	30%	40.00 TO 10 TO 10
test	虚机	500-2000	20%	0%-40%	20%	1-5	40%	20%			3編44 了贝贝。 即10水
m2>1 +0 mil	K8S	500-2000	10%	0%-40%	10%	1-5	40%	10%	1-5	30%	LAND 22114 MILLAN
款1 入规则	虚机	500-2000	20%	0%-40%	20%	1-5	40%	20%			骗挥 預见 副际

3.10.5.3. 健康检查

平台通过配置的 URL,对相应的 URL 进行健康度检测。

阈值设置 健康评分 健康检查 URL过滤								
相户 测试租户1 · · · · · 业务 System · · 时间 芭 1天								
需要进行健康检查的URL。								
URL列表 ①	根据实例或url查询 Q		选中URL列表	根据实例或url查询	٩			
- □ 全部实例			 ◆ 全部実例 ▲ ginx-ingress-controller ▲ default-http-backend 					



3.10.5.4. url 过滤

平台通过配置的 URL,对相应的 URL 进行过滤。

阈值设置 健康评分 健康检查 URL过滤				
相户 测试租户1 V 业务 System V				
指定需要被过滤的URL。配置生效后,过滤URL将不再出现在云监控系统中。				×
URL列表	根据实例或url查询 Q	过滤URL列表	根据实例或url查询	٩
 ◆ 全部实例 > nginx-ingress-controller > default-http-backend apm-abnormal-alarm mysql-mm-sts [fleet-agent coredns kube-proxy > alize satis 		 ◆ 全部交列 → nginx-ingress-controller > default-http-backend 		

3.10.6. 应用配置

此模块对 agent 探针相关进行参数配置,使探针可动态修改数据抓取的参数。

3.10.6.1. 应用恢复

可在此模块进行删除的应用或实例的恢复,配置后立即生效,支持批量操作。

租户: zzzzzzz		集群: k8-68	-dev 🗸 🗸							
应用恢复	应用阈值	URL重命名	健康指数	探针配置	异常过滤	外部服务配置				
批量恢复									根据实例名称搜索	
					应用名称		删除时间 ♀	操作人	操作	

3.10.6.2. 应用阈值

可在此处设置应用阈值,展示应用 topo 中的平均访问延时及超过访问时延的数据,配置 后立即生效。应用阈值页各栏支持对全部应用生效。



应用恢复	应用阈值	URL重命名	健康指数	探针配置	异常过滤	外部服务配置	භා
test		×]					
∨ 应用节点	响应时间阈值 ⊘						
应月	月阈值配置	0 1	ns				
是否生	效于全部应用						
保存配置							
✓ 慢数据采	集阈值						
慢事务	时间阈值 ⑦	100 n	ns				
		设置进行数据分析的	的性能阈值,单位为毫秒,缺	省值是500毫秒。当接口	响应时间超过该阈值时	,探针将会记录详细事务发生数据	
慢SQI	时间阈值 ⑦	500 n	ns				
		设置进行数据库存图	取的性能阈值,单位为毫秒,	缺省值是500毫秒。当S0	21响应时间超过该阈值8	时,探针将会记录详细的数据库存日	取超时数据。
数据库道	接超时阈值 ②	3000 n	ns				
		设置进行数据库连持	_{妾超时的性能阈值,单位为毫}	秒, 缺省值是3,000毫秒	。当数据库连接时间超	过该阈值时,探针将会记录详细的	数据库连接超时数据。
慢服务	调用阈值 ②	500 n	ns				
		设置进行数据分析的	的性能阈值,单位为毫秒,缺	省值是500毫秒。当接口	响应时间超过该阈值时	,探针将会记录详细服务发生数据	*
慢nc	osql阈值②	100 n	ns				
		设置进行数据库存	取的性能阈值,单位为毫秒, [;]	缺省值是100毫秒。当N(OSQI响应时间超过该阈	值时,探针将会记录详细的数据库	存取超时数据。
是否生	效士全部应用						
保存配置							

慢数据采集阈值包括: 慢事务时间阈值、慢 SQL 时间阈值、数据库连接超时阈值、慢服务 调用阈值、慢 nosql 阈值、慢消息队列超时阈值, 配置 2-4 分钟后在不同的页面生效

探针熔断阈值,用于配置探针熔断的条件。

探针熔断阈值 ②			6
老年代使用率	90	%	
noneheap使用率	90	%	
GC cpu使用率	100	%	
发送消息线程cpu使用率	95	%	
以上阈值连续达到次数	5		
是否生效于全部应用			
保存配置			

3.10.6.3. URL 重命名

可为事务重命名,便于区分辨认关键事务;配置后立即生效。



agent-demo		根据URL,名称搜索		٩
URL	自定义名称	操	ſĒ	
/exception		59		808
/n250_4		93		elloe
/sleep/*	睡眠	193	18 8	EURA
/user/*		49	HE 8	ellek

3.10.6.4. 探针配置

探针配置页用于各应用的 java 探针相关数据的配置。

296new	v
探针配置 ⑦	
是否解析http报文响应体 ②	✔ 勾选后解析http响应体内容
Java探针日志级别	Info V
	探针日志级别设置,等级为:off < error < warn < lnfo < debug ,修改后探针的日志级别将会动态生效
慢事务忽略包	org/apache/;weblogic/;org/jboss/;com/mchange/v2/;
	慢事务忽略包。缺省值为org/apache/;weblogic/;org/jboss/;com/mchange/v2/;org/springframework/;org/mybatis/;
忽略url列表	
	配置后,探针不抓取该url列表中的请求数据
慢方法配置	60
	应用探针定时探测惯方法。单位为ms,缺省值为60ms,若用户自定义时间间隔,则必须为60的倍数
	10
	缺省值为10。该配置项配置应用探针30分钟内重复抓取同一事务的最大次数。
	100
	单位为毫秒,缺省值500ms。当事务响应时间大于该阈值的时候,开始探测该事务中慢方法的数据。此设置必须大于定时探测慢方法间隔。
主动探测阈值(ms) ⑦	5000
	主动探测词值:如此词值大于慢方法词值,则开启主动探测模式;

3.10.6.5. 健康指数

健康指数配置页所配置的参数是计算应用的健康评分的基础。

Z用恢复	应用阈值	URL重命名	健康指数	探针配置	异常过滤	外部服务配置			
est									
③ 健康度总权重:0-	评分 = 响应时间 100整数,总和	可得分*响应时间权重+ 是100。计算公式:Ⅴ	+错误率得分*错误 表示实际值,M表;	郫权重+Apdex指数得分 示最大值,m表示最小	分*apdex权重 值。V<=m时,	分数=100;m <v<m时,分数=(m–v< td=""><td>)*100.0/(M-m); V>=M时 分数=0。</td><td></td><td></td></v<m时,分数=(m–v<>)*100.0/(M-m); V>=M时 分数=0。		
	*		最大值			最小	1	权重(此指	际占100分的比例)
Z时间 🕕		2000		n	ns	500	ms	40	%
£ ()		40			%	0	%	40	%



3.10.6.6. 异常过滤

异常过滤界面可配置对某类异常进行过滤。过滤之后,探针将不会抓取这一类的异常。

应用恢复 应	如用阈值 URL重命名	健康指数 探针配置	异常过	滤 外部服务配置		
应用: tes	it		URL:	URL.例:/index.php/user/login	时间范围:	
类名: 类名	8		方法名:	方法名	标记过滤:	
						查询
错误类型				•		详情
org.elasticsearch.E	ElasticsearchException					详情
org.elasticsearch.E	ElasticsearchStatusException					详情
org.springframewo	ork.web.bind.MissingServletRequest	tParameterException				详情
java.lang.NullPointe	terException					详情
java.net.ConnectEx	xception					详情
java.lang.IndexOut	tOfBoundsException					详情
cn.hutool.core.date	e.DateException					详情
org.elasticsearch.c	client.ResponseException					详情
java.io.IOException	n					详情
java.sql.SQLExcep	tion					详情
					第 1-10 条/总共 11条	< 1 2 > 10条/页 >

3.10.6.7. 外部服务配置

此模块可通过设置选择某些服务当作外部服务。

应用恢复	应用阈值	URL重命名	健康指数	探针配置	异常过滤	外部服务配	1				
内部服务					请输入搜索内容	٩		外部服务	请输入搜索内容	Q	+
elasticse	arch-svc:9200							10.1.30.50:6443			
10.10.1	1.69:30091										
10.10.1	02.160:30090										
10.10.1	01.107:80										
10.10.1	02.155:9090										
10.10.1	1.59:30090										
				*							

3.10.7. 组件监控

云监控平台是多个结构模块共同支撑起来的,对各个结构模块的状态检测就很有必要了。 在自监控模块的基础上,平台提供了一个展示各个模块状态的页面。



3.10.8. 磁盘清理

平台每时每刻都会有大量的数据产生,无限制的存储这些数据目前是有很大难度的,所以 必须对一些冷数据进行抛弃,只保留合理范围内的数据。平台采用的策略是根据时间线进行保存。如只保存最近7天的数据。

磁盘清理模块对 ES 索引的保留的时间设置, ES 将保留相对应时间的数据, 对超出时间的 冷数据进行抛弃。

集群: 测试用虚机集群 >>			
索引名称	数据库保留天数	磁盘大小	操作
JavaException	7	0	立即清理
npm_agg_topology	2	0	立即清理
channel	7	0	立即清理
kube-events	2	0	立即清理
jserror	7	0	立即清理
autodeploy	2	0	立即清理
agentheartbeat	7	0	立即清理
DBType	2	0	立即清理
npm_agg_entity	2	0	立即清理
npm_detail_topology	2	0	立即清理

点击清理按钮,可对超出设置时间的数据进行立即清理操作,与定时清理的功能相辅相成。






如需了解更多谐云信息 可通过扫描二维码或以下方式实时进行查阅