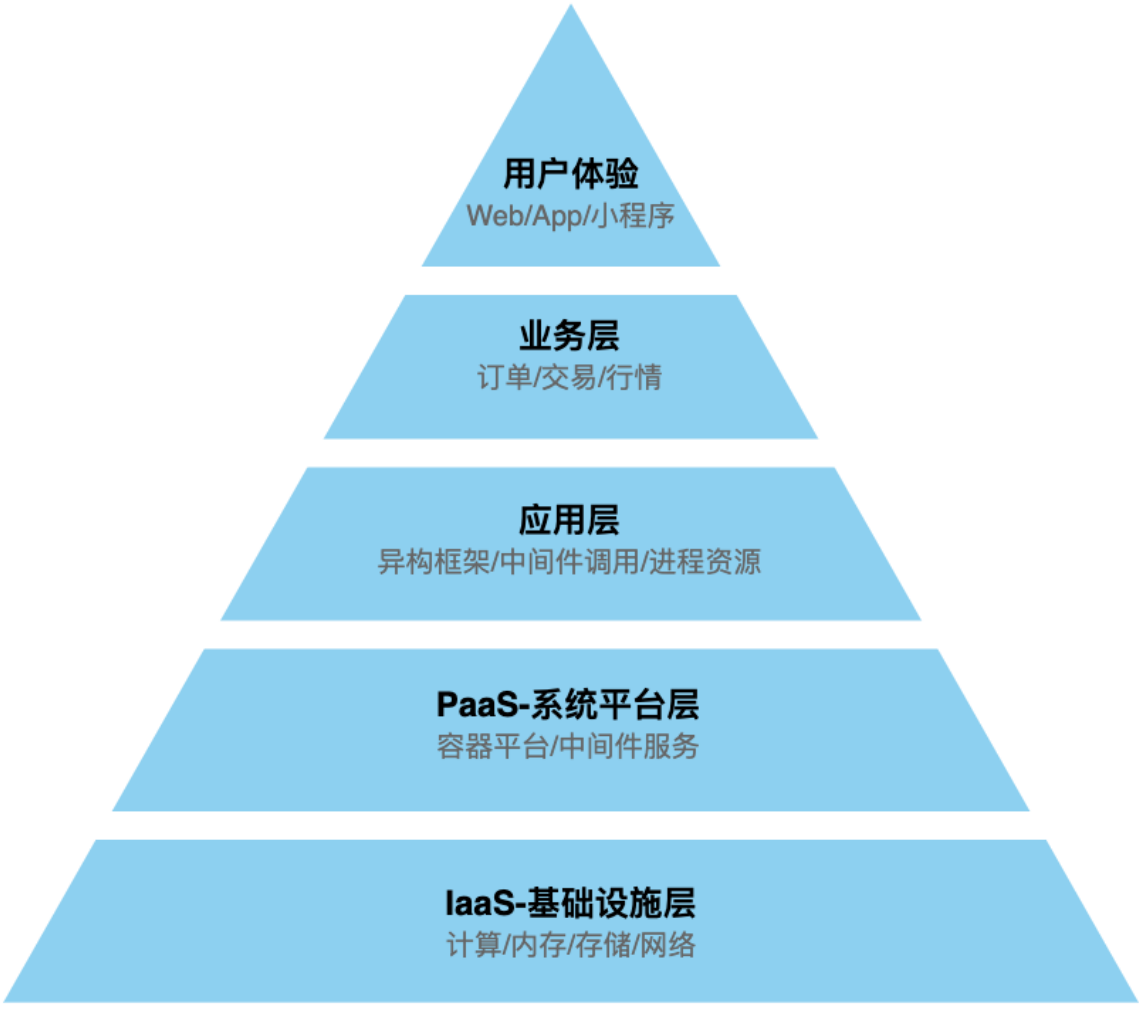
KOCA 监控运维平台概要设计-监控篇

目标

监控系统的目标是为了保障业务 SLA，全方位了解业务系统的运行状态，及时发现系统问题和潜在风险，为运维争取化解风险的时间，以及解决问题的方向。基于此监控体系需要以下核心能力：

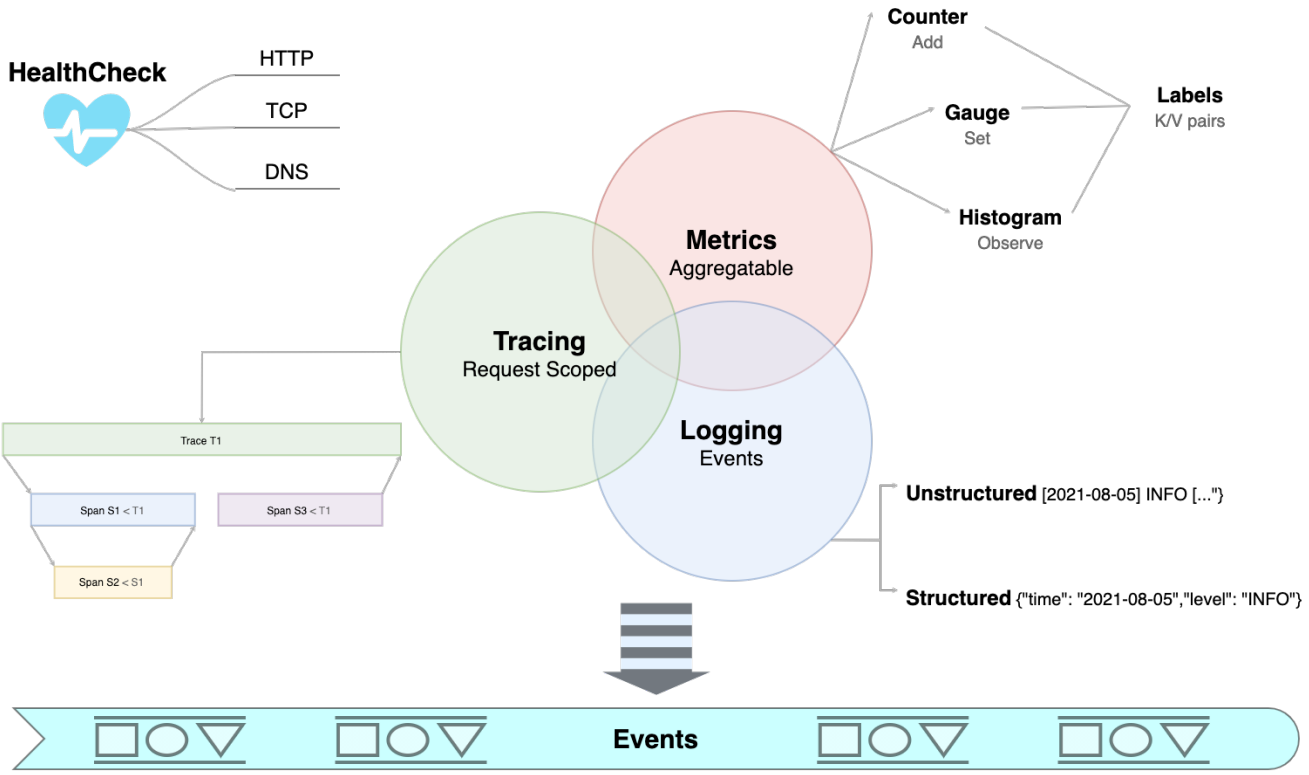
- **数据可视化：**通过监控系统获取的数据，可以生成可视化仪表盘、数据综合查询等，使运维人员能够直观地了解系统运行状态、资源使用情况、服务运行状态等。
- **趋势分析：**统计监控历史数据，对监控指标进行趋势分析。如：通过分析磁盘的使用空间增长率，可以预测何时需要对磁盘进行扩容。
- **对照分析：**随时掌握系统的不同版本在运行时资源使用情况的差异，或在不同容量的环境下系统并发和负载的区别。
- **告警：**当系统即将或已经出现故障时，监控可以迅速反应并发出告警。这样，就可以提前预防问题发生或快速处理已产生的问题，从而保证业务服务的正常运行。
- **故障分析与定位：**故障发生时，技术人员需要对故障进行调查和处理。通过分析监控系统记录的各种历史数据，可以迅速找到问题的根源并解决问题。
- **故障处理与跟踪**：从故障发生后需要完善的跟踪处理机制，包括故障记录、进度跟踪、各环节响应事件、定位原因、处理方案等等，包括系统”自愈“处理的问题，同样要做跟踪，这些都是系统持续改进的关键参考。

监控体系



监控系统可以分为：用户体验、业务层、应用层、PaaS 层和 IaaS 层。不同层级根据特点需要采用不同手段进行监控，并且要将各层的监控数据做关联分析，实现统一的监控平台。同时为不同用户角色提供多样化的能力，如：业务更关注数据的可视化，需要有多维数据展示的能力，仪表、大屏等；而运维更关注系统持续、稳定的运行以及故障的快速定位和处置能力，则需要可靠的故障分析和处理能力。作为监控运维平台首先要服务的还是偏技术一侧，而对于业务侧可以提供监控系统开放数据和服务的能力，供业务进行定制化应用。

监控方式



主要分为以下五类:

- **健康检查**：健康检查是对应用本身健康状况的监控，检查服务是否还正常运行；
- **度量指标**：指标是一些基于时间序列的离散数据点，通过聚合和计算后能反映出一些重要指标的趋势变化；
- **链路追踪**：链路追踪可以完整的呈现出一次请求的全部信息，包括服务调用链路、所耗时间、出错情况等；
- **日志**：日志是排查问题的主要方式，日志可以提供丰富的信息用于定位和解决问题；
- **事件**：事件是多种**关键**数据源的**汇总**，通过复杂事件处理(CEP)可以快速发现问题，还可以通过事件上下文发现系统**变更**情况，帮助定位和解决问题；

在上述 5 种监控方式中，健康检查是框架、平台等提供的能力；日志则通过日志中心进行日志的采集、转换、存储、计算和查询；链路追踪一般也有独立的解决方案进行服务调用的埋点、采集、计算和查询；指标监控则是通过一些 exporter 抓取目标暴露的指标，然后对这些指标数据进行清理、聚合，通过聚合出来的数据进行展示、告警等；事件相对比较零散，可以来自各个数据源的二次筛选，或者操作流水等日志记录，同样要经过采集、存储以及实时或离线的分析处理，最终根据处理结果进行告警、风险报告等

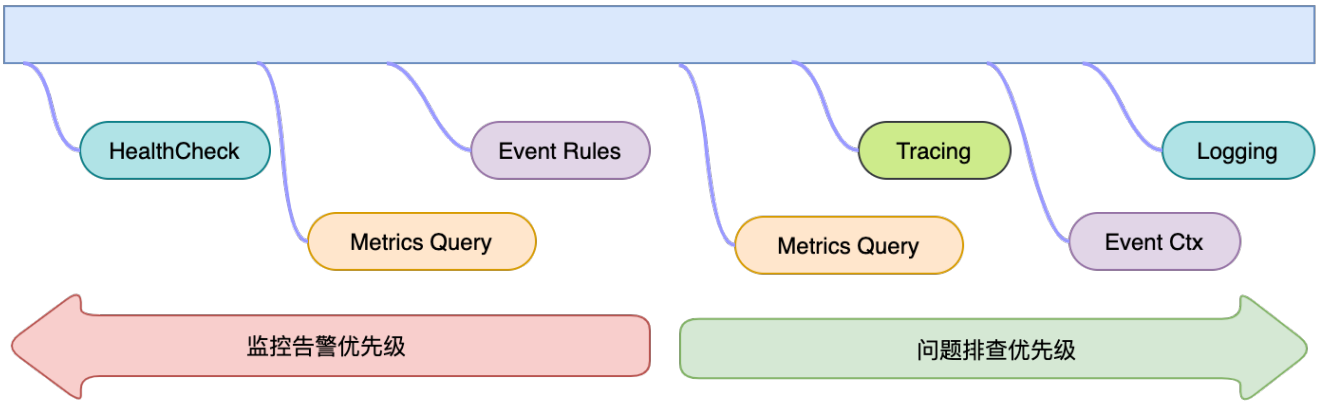
对于**事件**的引入可能会有些疑问，以下做个解释，一般从问题产生到反映在度量指标并产生告警会经过一个等待时间，故障发生后影响服务 -> 影响指标 -> 产生告警，而当一个或一组事件在系统中发生时，往往可以基于此判断系统即将发生问题。正如 ”海恩法则“ 所说的 ”一起重大的飞行安全事故背后都会有29个事故征兆“，事件监控就是系统在发生安全事故前通过这”29“个征兆来预判系统即将发生问题，虽然不能杜绝事故，但可以提早发现、干预或预防，至少为问题的解决赢得更多的时间和可能性。另外在问题定位阶段，日志往往是单一系统的，并且是直接问题

点，至于是什么导致的可能是配置的变更、运维操作、资源不足等，这些信息则散落在不同的服务中，如果每个有确定影响范围的事件都做一个集中管理，可以很好的辅助问题定位。

> 海恩法则：一起重大的飞行安全事故背后都会有29个事故征兆，每个征兆背后又有300个事故苗头，每个苗头背后还有1000个事故隐患。由此可见，对隐患、苗头、征兆的忽略，是导致意想不到的安全事故发生的罪魁祸首。——百度百科

应用场景

在监控的整个闭环反馈中不同的监控方式有不同的使用场景，从监控告警和问题排查两个维度看应用场景如下：



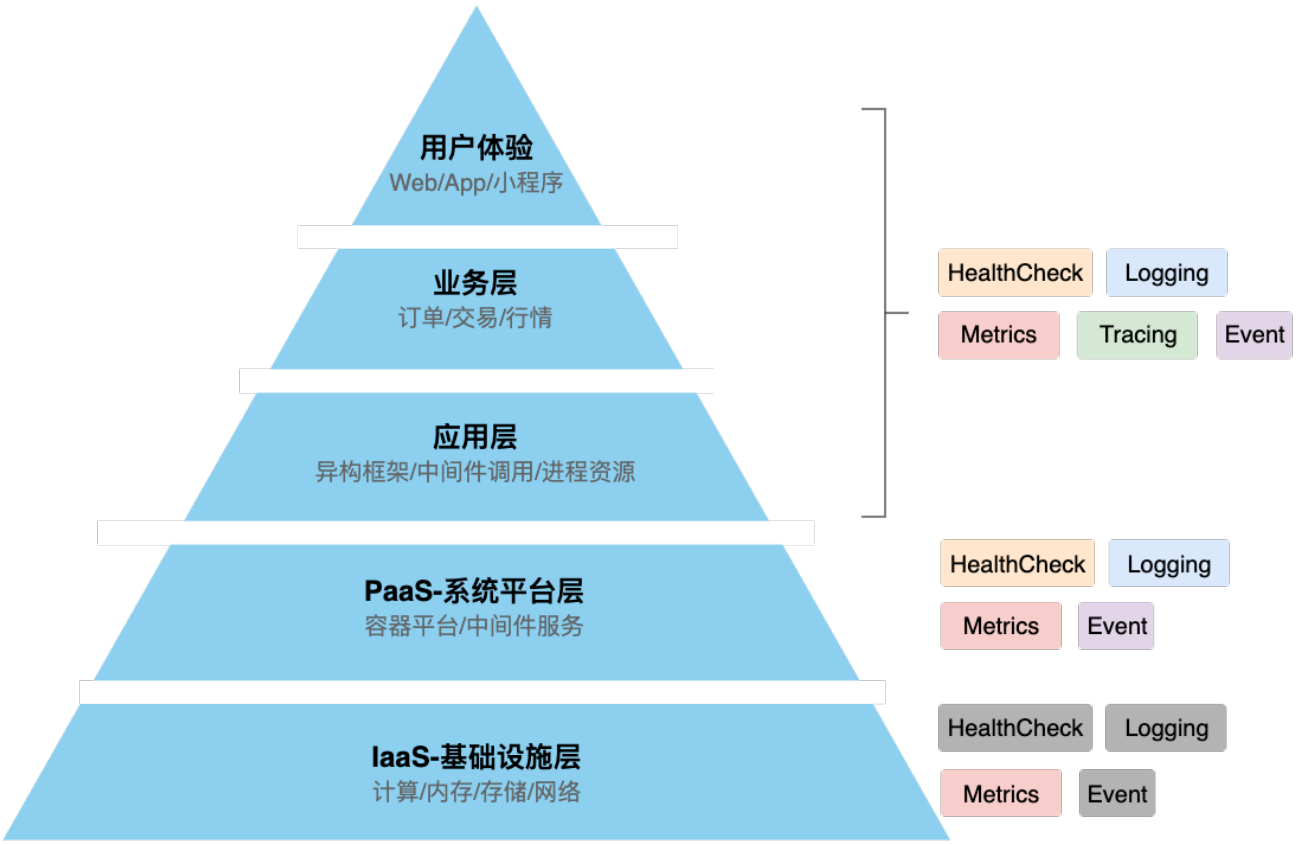
告警优先级为：**事件 > 度量指标 > 健康检查**

进行监控告警时，HealthCheck 是运维团队监测应用系统是否存活、是否健康的最后一道防线，这是必须引起重视的一道防线。HealthCheck 在微服务中通过对一个特定的 HTTP 请求进行轮询实现监控。通过对这个请求进行轮询，不但可以得到微服务的监控状态，还可以得到相关中间件如 MQ、Redis、MySQL、配置中心等的健康状态。当然，开发人员最为关心的监控还是自身定制的 Metrics 监控，所以监控告警的优先级依然是Metrics监控优先，HealthCheck 最低。由于 Event 的特点可以在早于 Metrics 发现问题，所以基于 Event 的告警优先级更高。

问题排查优先级为：**度量指标 > 链路 > 事件 > 日志**

Metrics 查询是基于时间序列的数据库设计得到的，可以直接定位到过去的任意时间点，可以对系统层、中间件层、应用层、业务层乃至终端上的所有监控指标进行查询。如果 Metrics 无法定位问题或者需要更多信息，Tracing 监控手段可以提供协助，帮助定位该问题发生在微服务链路的哪个环节（比如是物流服务、订单服务还是支付服务）。最后是使用事件上下文和日志，事件可以很好地打通关联系统，日志则可以定位系统内弄具体问题。通过 Metrics → Tracing → Event → Logging 的顺序分析问题，比直接去查日志更高效，很多问题都可以在日志之前的环节直接被定位并解决。

监控体系适用范围



不同监控方式与监控体系会有一个适用范围，从上图可以看到在上面三层基本所有的监控方式都可以用到，在 PaaS 层没有链路追踪，应用层与中间件的调用仅在应用层有发起请求是有 Client 端的调用链路，到中间件的 Server 端不再有链路追踪，在 IaaS 层可以用到的监控方式与 PaaS 基本一致，但在 AMO 平台没有关注这一层，比如各类系统日志、主机启停事件等等。

具体在各个监控体系可以做的哪些事情，以下是一个对应的表格：

	用户体验	业务层	应用层	PaaS / 中间件	IaaS
健康检查	站点是否正常访问	健康检查接口	健康检查接口，包括依赖服务	健康检查HTTP接口 / TCP / SSH	-
度量指标	站点访问的性能	业务指标	流量指标 资源占用的 Runtime 指标 中间件 Client 端指标	中间件 Server 端指标 容器平台指标	系统资源监控
链路追踪	终端请求与后端链路打通 终端以 Session、UID 等方式做到路径追踪	业务层链路埋点	服务间调用链路 中间件调用链路（同步/异步）	-	-
日志	JS 错误、接口错误、资源加载异常等 行为日志	业务日志	系统日志	慢日志 消息追溯	-
事件	埋点事件	业务事件	配置变更 运维操作 上下线、限流、熔断、调度等服务治理策略	容器平台事件	-

监控数据采集

健康检查

健康检查一般为黑盒监控，如 HTTP 接口、TCP 连接、端口是否正常、DNS、证书是否到期等，可以通过专用代理来完成不同的检查方式。健康监测的结果只有 0/1，并且是一个周期性进行的，存储上也是一个时间序列的离散数据，所以与度量指标有相同的技术方案。

度量指标

度量指标相对其它监控方式接入时最为复杂，度量指标需要根据被监控对象的特征做定量的指标设计，如：CPU负载、内存使用率、接口调用次数、平均响应时间等。常用的数据类型为计数器、测量仪和直方图，具体如何设计度量指标可以参考附录二：度量指标监控方法论，以及 AMO 基于 Prometheus 实践的指标规范，附录五：指标定义是 AMO 实践中的一些指标定义。一般采用两种主要方式完成指标采集：一是被监控对象内置指标数据，并能够提供不同协议的端点服务，如：Spring Boot Actuator 可以记录指标，并将指标以不同协议暴露；二是通过代理完成度量指标的捏去，并将指标以所需的协议暴露端点服务。现在一般现代的应用系统都会内置指标服务，以便于监控管理。

度量指标需要结合应用场景才能设计出合理的指标，看似复杂，不过在平台、框架层可以完成大部分工作，一般业务系统只需要关注业务层指标的设计即可。而且作为业务系统应该把度量指标的设计作为系统设计的一部分，在软件的整个生命周期中都要关注，需要认真设计，因为指标是否合理往往直接影响系统上线后发现问题和解决问题的效率，一个好的指标应该能够用于预测或定位问题，或是从业务的角度能够反映业务状态、趋势等。

以上是度量指标的生产端，追踪这些数据要能够采集到的时序数据库做持久化存储，并对外提供查询服务，采集一般分为两种模式：一是拉模式，即由采集服务通过被监控对象所暴露的端点服务完成指标的抓取；二是推模式，即由被监控对象的生产端主动上报数据到监控服务。

链路追踪

链路追踪主要针对的是微服务的分布式部署场景，如何实现调用链的追踪，基本都是基于 Google 的论文《Dapper, a Large-Scale Distributed Systems Tracing Infrastructure》，行业常用标准为 OpenTracing，调用链在请求上下文中传递 Tracing 信息，并在需要追踪的节点记录 Span 信息，最终再将散落在各个节点的 Span 信息进行汇总，通过 Traceing 上下文将整个链路串联起来，形成一个基于单个请求的完整调用链路。有关 Tracing 以及监控标准更多内容参考附录三：OpenTracing 标准、附录四：OpenTelemetry 标准 。链路的采集一般服务间以及服务于中间件间(同步/异步)的链路追踪可以由框架层来解决，而业务层的链路则需要业务开发过程中进行手动埋点。产生的链路信息一般可以通过采集器推送到链路服务，或者以 HTTP 接口等方式直接推送到链路服务，可以根据场景不同进行选择，前一种方式如果链路的存储服务与日志一致也可以与日志走相同的技术路径。

日志

日志是了解系统运行情况最简单的方法，也是最传统的，基本所有系统都有，正因此日志的规范化治理一般不是在生产端，而是在采集的中间环节加入转换、清晰过程，将纯文本日志转换为结构化的日志，为数据的应用做好基础。虽然日志可以在采集过程中进行转换清晰，不过一个标准的日志格式也可以大大简化处理过程的复杂度，因此给出一些实践参考附录一：日志记录实践。

事件

事件监控有别于其它几种监控方式，它是多渠道数据的汇总，既有源数据，也有加工处理后的数据。源数据如：修改配置、服务启停、容器平台的 Event；加工处理后的数据如：告警、清洗过滤的特殊日志/链路等。所有可能引起系统状态变化的行为，或是有助于发现问题、定位问题的操作、变更、信息都可以作为事件记录下来，一些特定的低频事件可以直接出发告警（如：容器平台调度失败）。事件的记录本身就是系统状态变化的"**关键**"信息，所以在定位问题时往往可以直接基于这些事件的上下文定位到问题原因。

监控方式对比

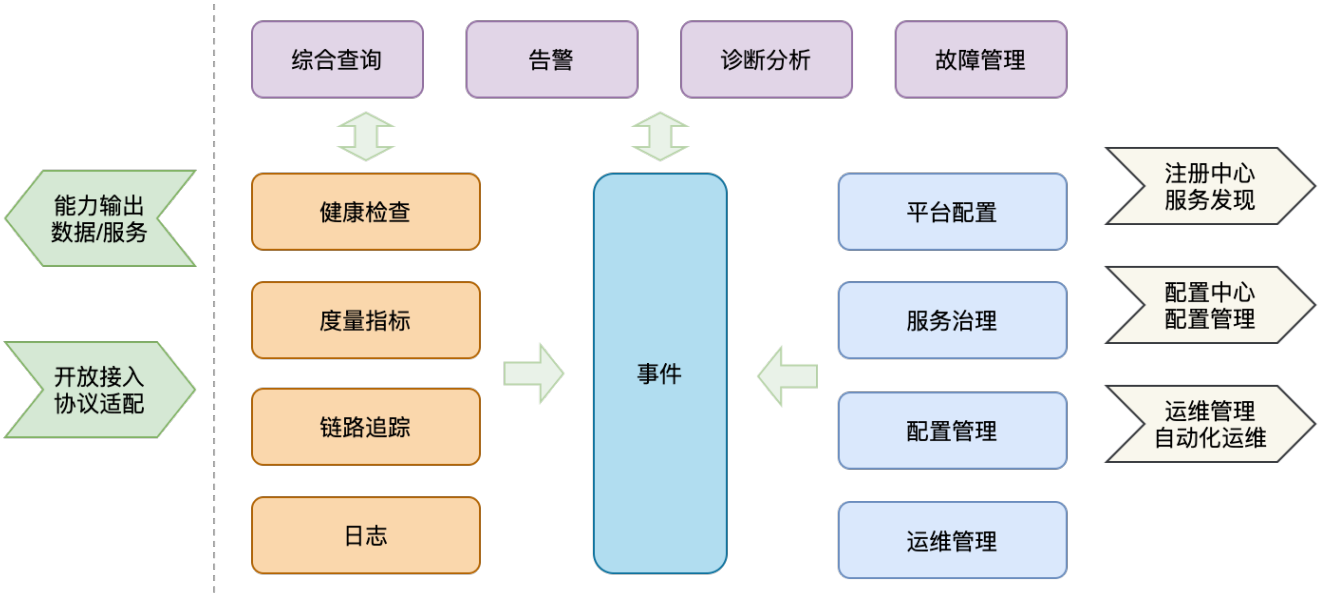
主要从以下四个维度对不同的监控方式进行对比：

- CapEx代表搭建的投入成本
- OpEx代表运维成本
- Reaction代表监控手段的响应能力
- Investigation代表查问题的有效程度

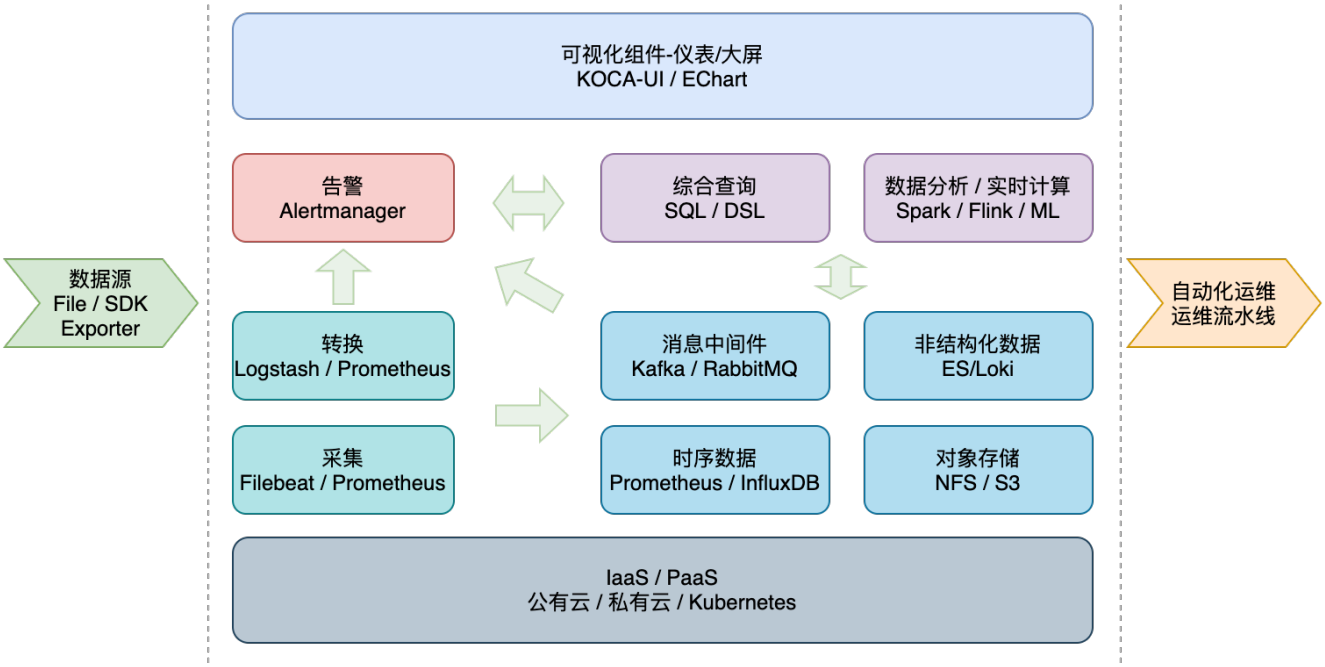
	Metrics	Logging	Tracing	Healthcheck	Event
CapEx-初始成本	中	低	高	低	中
OpEx-持续成本	低	高	中	低	中
Reaction-发现问题速度	高	中	低	高	高
Investigation-解决问题能力	低	中	高	-	中

架构设计

业务架构



技术架构



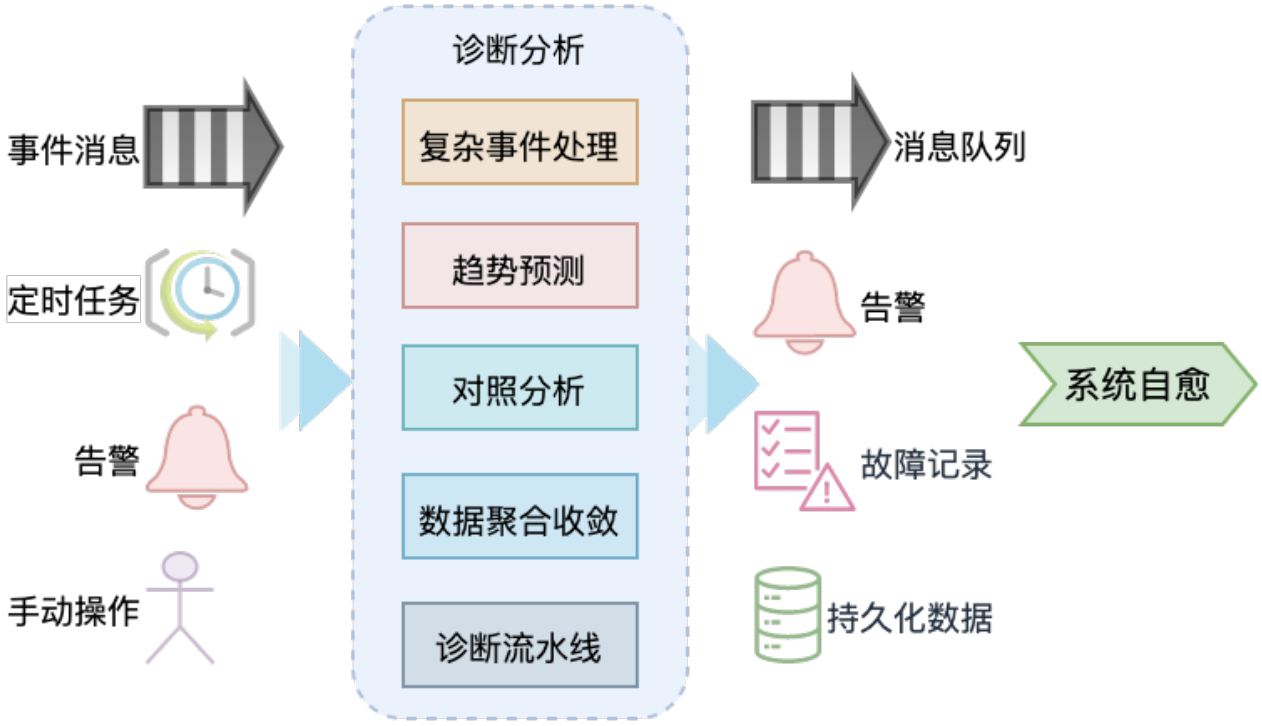
> 分层架构
>
> - 开放层
>   - 输出数据或服务能力
> - 应用层
>   - 平台管理、界面交互等
> - 处理层

    - 预处理、存储、统计查询、实时/离线计算
- 接入层
    - 协议适配，数据规范、校验
- 数据源
    - SDK、Exporter 等

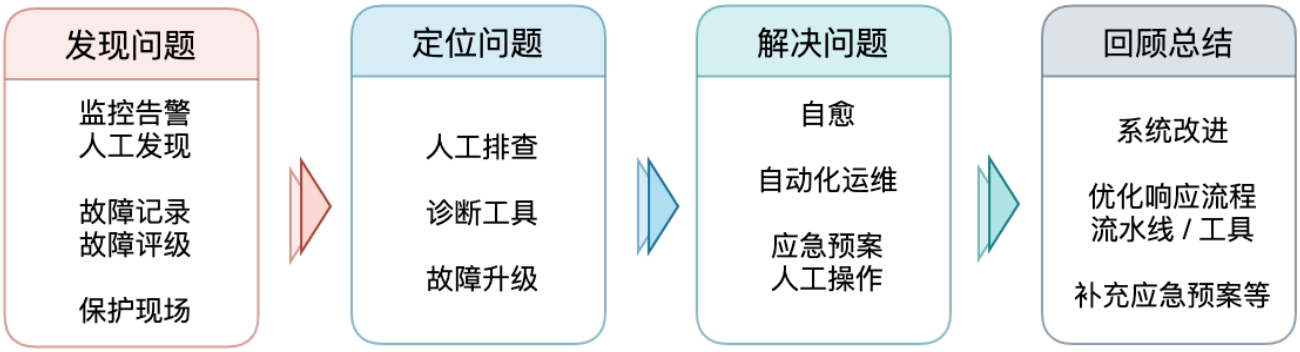
诊断分析

诊断分析能力有别于人工诊断，是将个人经验转化为代码、流程的过程，使运维能力可以被继承、维护和分享。诊断分析是以监控数据为基础，结合运维、运行时管理等手段的一个综合服务，即要负责故障的发现与预测，也要辅助和简化问题定位。

拆开来看就是诊断和分析，分析是要提供实时或离线的数据分析能力，包括基于事件流的复杂事件处理(CEP)，基于度量指标的趋势分析和预估、性能分析、以及对照分析等，同时还有数据的收敛和聚合；诊断是基于脚本、流水线的自动化诊断过程，包括监控数据查询、实例操作、JVM 诊断工具等，最终输出告警、事件、故障记录、故障现场等结果。其中故障现场可以理解为故障发生时完整的系统状态，试想在产生一条告警后触发一个流水线，完成上下游服务的状态、进程/JVM资源、运行时配置、近期事件上下文等数据的抓取和存储，有了这些数据再运维人员接到告警后可以第一时间基于这些信息完成问题的定位，或是再进一步可以基于此自动化的完成一些**自愈**的运维操作，如扩缩容、流量调度等。



故障管理



故障管理分为四个阶段：

- **发现问题：**在系统平稳运行阶段通过告警、人工、数据分析等手段发现问题，根据情况自动或是手动的完成故障记录以及故障定级，并分派责任人，有关故障定级可参考[附录七：故障等级划分。此过程要完成的另一个重要工作是“保护现场”，故障发生时的系统状态对排查问题很有帮助，可以通过多种手段对故障信息进行补充，如：当前配置，相关链路、日志，系统资源，进程资源，相关事件等等。
- **定位问题：**根据问题情况可以进行自动或人工定位分析，可以自动解决的无序人工干预，根据策略触发自愈过程，需要人工排查的通过各项监控项以及现场数据，对问题进行定位分析，如果超出责任或能力范围的对故障进行升级，求得更多资源。
- **解决问题：**可自愈问题自动完成自愈操作，如：清理磁盘，重启实例，流量控制等。其它则根据故障情况通过自动化运维过程或其它应急方式决绝问题。
- **回顾总结：**此过程主要是对故障系统进行整改，可能是一个持续的过程，但对于运维人员可以针对故障响应流程、诊断工具、文档等方面进行总结，并做相应的优化，补充完善应急预案等。

监控平台除提供监控运维相关的能力外，还要提供故障跟踪系统，有标准的故障记录模型，以及全链路跟踪的系统支持，包括故障相关诊断数据的归档，定位、解决问题的过程、方法等。

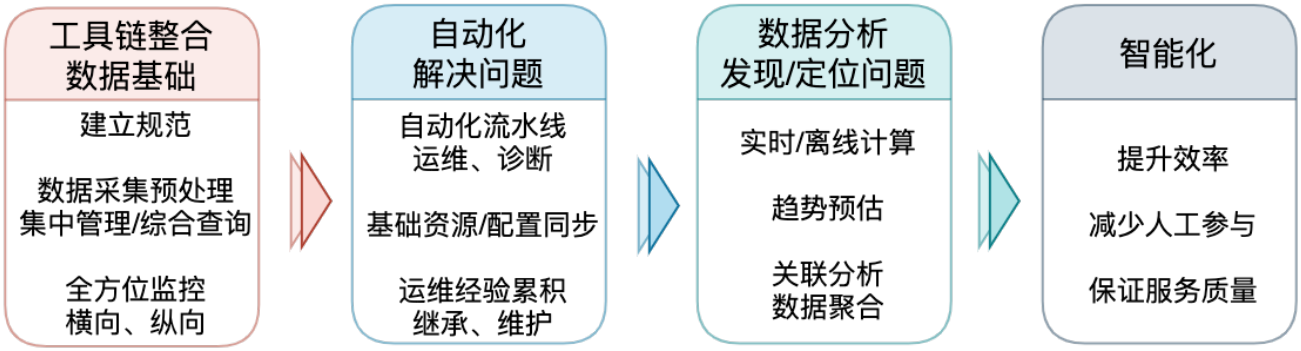
变更管理

引发故障或异常最多的往往不是软件BUG，而是各类“变更”所导致的，如配置修改、实例启停、流量调度、告警规则、指标配置等等，要做好变更管理可以从以下几个方面做起：

- 需要有规范的配置模型，可以对低级错误进行校验，格式校验、完整性、值范围等
  - 包括配置间的关联性进行合理、合规方面的检测
- 配置的变更与生产做隔离，至少有个版本管理和发布过程
- 能够对配置发布后的生效情况进行跟踪，如果下发或者使用方报错可以给出反馈，确保配置的一致性
- 对于影响范围广的配置做灰度发布，确认没有问题后在全量发布
- 在有条件的环境下可以做沙盒验证，经过验证后再发布到生产
- 特殊配置需要引入必要的复核、审批流程

以上是对配置变更管理的一些基本要求，这些同样可以映射到运维的脚本、计划、流程等管理过程，所有的“变更”都要有明确的影响范围，并且将变更记录加入到事件监控，作为发现和解决问题的重要信息源。

建设过程



附录

附录一：日志记录最佳实践

1. 日志记录有详细的时间戳，年月日时分秒毫秒，建议使用yyyy-MM-dd HH:mm:ss.SSS这样的格式；
2. 不同字段间有明确的分隔符，便于正则解析，如使用“空格”分隔，空数据使用“-”站位，字段信息内有空格的可以使用“”、[]等进行分隔，注意一般日志最后的内容部分不需要这样处理；
3. 多行日志要能够区分首行，一般首行能够通过正则匹配，而换行后可以给出空格等方式避免与首行正则冲突；
4. 日志分级，KOCA平台建议使用FATAL/ERROR/WARN/INFO/DEBUG分级，如果已有系统并没有分级或是分级方式不同，在ETL过程最好能够映射为统一日志级别，便于日志的筛选与统计；
5. 日志隔离，不同类型日志通过文件目录或文件名进行区分，便于采集时区分日志类型，对应koca_log.type;
6. 编码格式最好使用UTF-8，如果不是在采集端配置编码，如Filebeat配置encoding=GBK，避免中文乱码的产生;
7. 带有链路追踪的系统建议在日志中增加链路ID标识，便于链路日志的钻取，排查问题；

附录二：度量指标监控方法论

Google 的四大黄金指标

有4个来自Google SRE手册的黄金指标，这4个指标主要针对应用程序或用户部分。

- **延迟（Latency）**：服务请求所需耗时，例如HTTP请求平均延迟。需要区分成功请求和失败请求，因为失败请求可能会以非常低的延迟返回错误结果。
- **流量（Traffic）**：衡量服务容量需求（针对系统而言），例如每秒处理的HTTP请求数或者数据库系统的事务数量。
- **错误（Errors）**：请求失败的速率，用于衡量错误发生的情况，例如HTTP 500错误数等显式失败，返回错误内容或无效内容等隐式失败，以及由策略原因导致的失败（比如强制要求响应时间超过30ms的请求为错误）。
- **饱和度（Saturation）**：衡量资源的使用情况，例如内存、CPU、I/O、磁盘使用量（即将饱和的部分，比如正在快速填充的磁盘）。

Netflix 的 USE 方法

USE是Utilization（使用率）、Saturation（饱和度）、Error（错误）的首字母组合，是Netflix的内核和性能工程师Brendan Gregg提出的，主要用于分析系统性能问题，可以指导用户快速识别资源瓶颈及错误。

- **使用率**：关注系统资源的使用情况。这里的资源主要包括但不限于CPU、内存、网络、磁盘等。100%的使用率通常是系统性能瓶颈的标志。

- **饱和度：**例如CPU的平均运行排队长度，这里主要是针对资源的饱和度（注意，不同于四大黄金指标）。任何资源在某种程度上的饱和都可能导致系统性能的下降。
- **错误：错误数。**例如，网卡在数据包传输过程中检测到以太网络冲突了14次。

3Weave Cloud 的 RED 方法

RED 方法是 Weave Cloud 基于 Google 的4个黄金指标再结合 Prometheus 及 Kubernetes 容器实践得出的方法论，特别适用于对云原生应用以及微服务架构应用进行监控和度量。在四大黄金指标的原则下，RED方法可以有效地帮助用户衡量云原生以及微服务应用下的用户体验问题。RED方法主要关注以下3种关键指标。

- **（Request）Rate：**每秒接收的请求数。
- **（Request）Errors：**每秒失败的请求数。
- **（Request）Duration：**每个请求所花费的时间，用时间间隔表示。

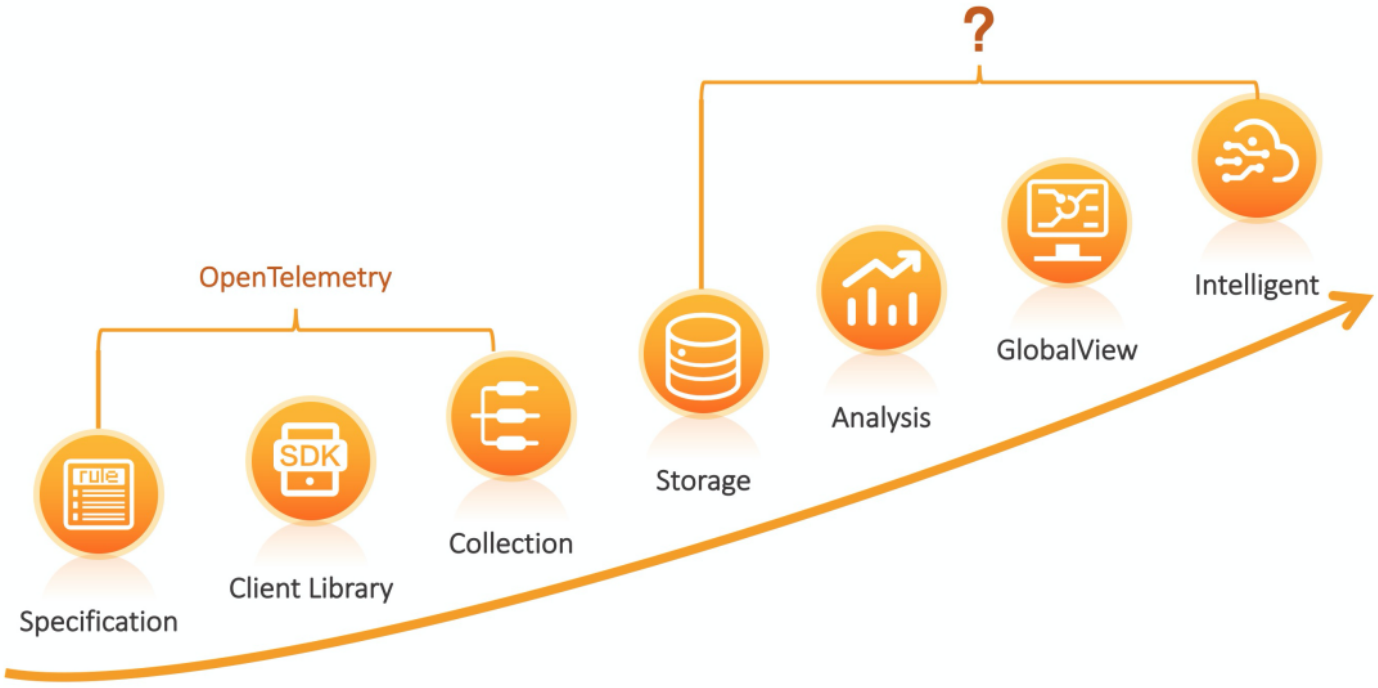
一般来说，上述三大监控理论的最佳实践是：在遵循Google四大黄金指标的前提下，对于在线系统，结合RED方法和缓存命中率方式进行监测；对于离线系统或者主机监控，以USE方法为主进行监测；对于批处理系统，可以采用类似Pushgateway的形式进行监控。

附录三：OpenTracing 标准

> [OpenTracing] (https://opentracing.io/)
>
> 已合并到 OpenTelemetry

附录四：OpenTelemetry 标准

日志、链路、度量已经成为微服务系统可观测性的三大保证， 有各种各样的开源或商业产品，有的关注某个领域，有的能够兼顾，但没有一致性的标准，尤其链路和度量最为突出，当前在开源界做有望将标准统一的是 OpenTelemetry，各大云厂的监控系统都在做这方面的支持，平台框架也都在向标准靠拢。OpenTelemetry 标准主要关注在Logs、Traces 和 Metrics 的结构，以及数据采集过程，下游的存储、分析、展示等则是对接其它产品。



附录五：指标定义

> TODO 根据相应标准以及 KOCA 实践完善指标体系

应用层

RPC 服务调用

指标

指标	类型	说明
koca_requests_total	counter	请求计数
koca_request_duration_seconds_max	gauge	请求最大响应时间
koca_request_duration_seconds	histogram	请求响应时间 `0.01,0.02.0.05,0.1,0.2,0.5,1,3,8,20,60,120`
koca_request_bytes	histogram	请求流量 `100,1000,10000,100000,1000000,10000000,100000000,1000000000`
koca_response_bytes	histogram	响应流量 `100,1000,10000,100000,1000000,10000000,100000000,1000000000`

标签

> *不同协议可以根据情况增减标签*

标签	说明
repoter	`destination` 或 `source` Inbound 流量 `reporter=distination`， `source_service` 为下游服务， `destination_service` 为当前服务； Outbound 流量 `reporter=source`， `source_service` 为当前服务， `destination_service` 为上游服务
code	协议的响应错误码，如 HTTP 的 `200` `404` 等
uri	`http.URI`
method	`http.Method`
koca_code	KOCA 接口错误码
protocol	协议 `http` 、 `grpc` 等
source_service	源服务名称
source_version `options`	源服务版本号
destination_service	目标服务名称
destination_version `options`	目标服务版本号

JVM

key	值类型	说明
jvm_buffer_memory_used_bytes	gauge	
jvm_classes_unloaded_classes_total	counter	
jvm_threads_live_threads	gauge	
jvm_memory_committed_bytes	gauge	
jvm_memory_used_bytes	gauge	
jvm_threads_peak_threads	gauge	
jvm_buffer_total_capacity_bytes	gauge	
jvm_gc_memory_allocated_bytes_total	counter	
jvm_threads_daemon_threads	gauge	
jvm_memory_max_bytes	gauge	
jvm_threads_states_threads	gauge	
jvm_buffer_count_buffers	gauge	
jvm_gc_live_data_size_bytes	gauge	
jvm_gc_pause_seconds	summary	
jvm_gc_pause_seconds_max	gauge	
jvm_classes_loaded_classes	gauge	
jvm_gc_max_data_size_bytes	gauge	
jvm_gc_memory_promoted_bytes_total	counter	
tomcat_sessions_active_current_sessions	gauge	
tomcat_sessions_active_max_sessions	gauge	
tomcat_sessions_expired_sessions_total	counter	
tomcat_sessions_rejected_sessions_total	counter	
tomcat_sessions_created_sessions_total	counter	
tomcat_sessions_alive_max_seconds	gauge	

中间件 Client 端

> TODO
>
> - DB
> - MQ
> - Cache

业务层

- 报盘网关

> `交易所` `席位` `接口类型` 组合的基数?

key	值类型	标签	说明	问题
start_time_milliseconds	gauge		启动时间	
mem_cache_size	gauge		Cache 占用内存	
seat_num	gauge		席位数量	
multicast_num	gauge		组播序号	
seat_status	gauge	`交易所` `席位` `接口类型`	席位状态	委托路径是否需要?
seat_assign_num	counter	`交易所` `席位` `接口类型`	委托数量	
seat_accept_num	counter	`交易所` `席位` `接口类型`	确认数量	
seat_deal_num	counter	`交易所` `席位` `接口类型`	成交数量	

- 交易核心

key	值类型	标签	说明	问题
start_time_milliseconds	gauge		启动时间	
mem_cache_size	gauge		Cache 占用内存	
mem_db_size	gauge	`db_name`	数据库总大小	
mem_db_used	gauge		数据库已使用	
lbm_total_num	gauge		后台LBM总数	
tcp_conn_num	gauge		TCP连接数	
curr_business_num	gauge		当前业务序号	
trans_thread_num	gauge		交易线程数量	
risk_management_num	gauge		风控线程数量	
lbm_request_total	counter	`function_num`	总调用数	功能号基数?
lbm_success_total	counter	`function_num`	成功次数	
lbm_fail_total	counter	`function_num`	失败次数	
lbm_first_fail_num	gauge	`function_num`	首次失败序号	
lbm_business_duration_milliseconds	gauge/histogram	`function_num`	业务执行耗时	
lbm_response_duration_milliseconds	gaugehistogram	`function_num`	应答等待耗时	
lbm_duration_milliseconds	gaugehistogram	`function_num`	整体执行耗时	
lbm_duration_min_milliseconds	gauge	`function_num`	最小运行时间	是否可以用运行时间替换
lbm_duration_max_milliseconds	gauge	`function_num`	最大运行时间	是否可以用运行时间替换

- 交易网关

key	值类型	标签	说明	问题
mem_cache_size	gauge		Cache 占用内存	
start_time_milliseconds	gauge	`网关ID` `缓存名称` `TCP监听地址` `组播接受序号` `组播发送序号`		
kcxp_send_thread_num	gauge		KCXP发送线程数	
kcxp_receive_thread_num	gauge		KCXP接收线程数	

- 消息中心

key	值类型	标签	说明	问题
mem_cache_size	gauge		Cache 占用内存	
start_time_milliseconds	gauge	RabbitMQ `输入IP` `输出IP` `输入端口` `输出端口`	启动时间，含标签信息	
	gauge	`IP地址` `输入端口`	组播接收序号	
	gauge		转换业务调用间隔	
	gauge		转换业务调用池中线程	
	gauge		通用业务调用池中线程	
	counter	`消息类型` `消息ID`	消息已发送数量	消息ID是否可枚举?
	gauge	`消息类型` `消息ID`	最后发送时间	

- 行情网关

key	值类型	标签	说明	问题
mem_cache_size	gauge		Cache 占用内存	
	gauge		最近接收数量	
	gauge		最近更新数量	
	gauge		最近接收时间	
	gauge		最近更新时间	
	gauge	`行情连接地址`	程序运行状态	

- FIX网关

key	值类型	标签	说明	问题
mem_cache_size	gauge		Cache 占用内存	
start_time_milliseconds	gauge	`组播序号`	启动时间	组播序号是否变动?
	gauge		连接数量	
	gauge	`名称`	交易状态	名称是否可枚举?
	gauge/counter		委托数量	数量是状态值还是累计计数
	gauge/counter		执行数量	
	gauge/counter		撤单数量	
	gauge/counter		改单数量	
	gauge/counter		撤单拒绝数量	
	gauge/counter		委托查询数量	
	gauge/counter		业务拒绝数量	
	gauge		登录时间	
	gauge		登出时间	

中间件

- KCBP

key	值类型	标签	说明	问题
kcbp_appid	gauge	`appid`		是否为BP节点序号或两者关系?
mem_cache_size	gauge		Cache 占用内存	
kcbpas_current_state	gauge	`index`	kcbpas当前状态，`0/1`	
kcbpas_processed_request	gauge	`index`	kcbpas处理请求数	是否为累计值?
kcxp_mem_db_num	gauge	`db_name`	内存库表数量	
kcxp_mem_db_size	gauge		内存库总大小	
kcxp_mem_db_used	gauge		内存库已使用	
kcxp_lbm_num	gauge		LBM数量	
kcxp_lbm_maxcost	gauge	`lbmname`	LBM最大耗时	LBM名称的基数?
kcxp_lbm_refer_total	gauge	`lbmname`	LBM调用次数	
kcxp_lbm_timecost	gauge	`lbmname`	LBM总耗时	累计值? 是否可以用响应时间替换
kcxp_lbm_duration_milliseconds	gauge/histogram/summary	`lbmname`	LBM响应时间	

- KCXP

> 队列名称的基数?

key	值类型	标签	说明	问题
mem_cache_size	gauge		Cache 占用内存	
kcxp_queue_push_total	counter	`queue_name`	进队列消息数	
kcxp_queue_pop_total	counter		出队列消息数	
kcxp_queue_deep	gauge		队列深度	深度预警值是一个报警值还是应用内配置的变量
kcxp_queue_deep_max	gauge		队列最大深度	历史最大深度？是否可以用`max(deep)`替代？

> TODO
>
> - Oracle
> - MySQL
> - SQLServer
> - Kafka
> - Redis

容器平台

Kubernetes 集群

> TODO

基础设施

- Linux 主机

key	值类型	说明
node_arp_entries	gauge	
node_boot_time_seconds	gauge	
node_context_switches_total	counter	
node_cooling_device_cur_state	gauge	
node_cooling_device_max_state	gauge	
node_cpu_guest_seconds_total	counter	
node_cpu_seconds_total	counter	
node_disk_io_now	gauge	
node_disk_io_time_seconds_total	counter	
node_disk_io_time_weighted_seconds_total	counter	
node_disk_read_bytes_total	counter	
node_disk_read_time_seconds_total	counter	
node_disk_reads_completed_total	counter	

node_disk_reads_merged_total	counter	
node_disk_write_time_seconds_total	counter	
node_disk_writes_completed_total	counter	
node_disk_writes_merged_total	counter	
node_disk_written_bytes_total	counter	
node_entropy_available_bits	gauge	
node_exporter_build_info	gauge	
node_filefd_allocated	gauge	
node_filefd_maximum	gauge	
node_filesystem_avail_bytes	gauge	
node_filesystem_device_error	gauge	
node_filesystem_files	gauge	
node_filesystem_files_free	gauge	
node_filesystem_free_bytes	gauge	
node_filesystem_readonly	gauge	
node_filesystem_size_bytes	gauge	
node_forks_total	counter	
node_intr_total	counter	
node_load1	gauge	
node_load15	gauge	
node_load5	gauge	
node_memory_Active_anon_bytes	gauge	
node_memory_Active_bytes	gauge	
node_memory_Active_file_bytes	gauge	
node_memory_AnonHugePages_bytes	gauge	
node_memory_AnonPages_bytes	gauge	
node_memory_Bounce_bytes	gauge	
node_memory_Buffers_bytes	gauge	
node_memory_Cached_bytes	gauge	

node_memory_CmaFree_bytes	gauge	
node_memory_CmaTotal_bytes	gauge	
node_memory_CommitLimit_bytes	gauge	
node_memory_Committed_AS_bytes	gauge	
node_memory_DirectMap1G_bytes	gauge	
node_memory_DirectMap2M_bytes	gauge	
node_memory_DirectMap4k_bytes	gauge	
node_memory_Dirty_bytes	gauge	
node_memory_HardwareCorrupted_bytes	gauge	
node_memory_HugePages_Free	gauge	
node_memory_HugePages_Rsvd	gauge	
node_memory_HugePages_Surp	gauge	
node_memory_HugePages_Total	gauge	
node_memory_Hugepagesize_bytes	gauge	
node_memory_Inactive_anon_bytes	gauge	
node_memory_Inactive_bytes	gauge	
node_memory_Inactive_file_bytes	gauge	
node_memory_KernelStack_bytes	gauge	
node_memory_Mapped_bytes	gauge	
node_memory_MemAvailable_bytes	gauge	
node_memory_MemFree_bytes	gauge	
node_memory_MemTotal_bytes	gauge	
node_memory_Mlocked_bytes	gauge	
node_memory_NFS_Unstable_bytes	gauge	
node_memory_PageTables_bytes	gauge	
node_memory_SReclaimable_bytes	gauge	
node_memory_SUnreclaim_bytes	gauge	
node_memory_Shmem_bytes	gauge	

node_memory_Slab_bytes	gauge	
node_memory_SwapCached_bytes	gauge	
node_memory_SwapFree_bytes	gauge	
node_memory_SwapTotal_bytes	gauge	
node_memory_Unevictable_bytes	gauge	
node_memory_VmallocChunk_bytes	gauge	
node_memory_VmallocTotal_bytes	gauge	
node_memory_VmallocUsed_bytes	gauge	
node_memory_WritebackTmp_bytes	gauge	
node_memory_Writeback_bytes	gauge	
node_netstat_Icmp6_InErrors	untyped	
node_netstat_Icmp6_InMsgs	untyped	
node_netstat_Icmp6_OutMsgs	untyped	
node_netstat_Icmp_InErrors	untyped	
node_netstat_Icmp_InMsgs	untyped	
node_netstat_Icmp_OutMsgs	untyped	
node_netstat_Ip6_InOctets	untyped	
node_netstat_Ip6_OutOctets	untyped	
node_netstat_IpExt_InOctets	untyped	
node_netstat_IpExt_OutOctets	untyped	
node_netstat_Ip_Forwarding	untyped	
node_netstat_TcpExt_ListenDrops	untyped	
node_netstat_TcpExt_ListenOverflows	untyped	
node_netstat_TcpExt_SyncookiesFailed	untyped	
node_netstat_TcpExt_SyncookiesRecv	untyped	
node_netstat_TcpExt_SyncookiesSent	untyped	
node_netstat_TcpExt_TCPSynRetrans	untyped	
node_netstat_Tcp_ActiveOpens	untyped	
node_netstat_Tcp_CurrEstab	untyped	

node_netstat_Tcp_InErrs	untyped	
node_netstat_Tcp_InSegs	untyped	
node_netstat_Tcp_OutSegs	untyped	
node_netstat_Tcp_PassiveOpens	untyped	
node_netstat_Tcp_RetransSegs	untyped	
node_netstat_Udp6_InDatagrams	untyped	
node_netstat_Udp6_InErrors	untyped	
node_netstat_Udp6_NoPorts	untyped	
node_netstat_Udp6_OutDatagrams	untyped	
node_netstat_Udp6_RcvbufErrors	untyped	
node_netstat_Udp6_SndbufErrors	untyped	
node_netstat_UdpLite6_InErrors	untyped	
node_netstat_UdpLite_InErrors	untyped	
node_netstat_Udp_InDatagrams	untyped	
node_netstat_Udp_InErrors	untyped	
node_netstat_Udp_NoPorts	untyped	
node_netstat_Udp_OutDatagrams	untyped	
node_netstat_Udp_RcvbufErrors	untyped	
node_netstat_Udp_SndbufErrors	untyped	
node_network_address_assign_type	gauge	
node_network_carrier	gauge	
node_network_carrier_changes_total	counter	
node_network_device_id	gauge	
node_network_dormant	gauge	
node_network_flags	gauge	
node_network_iface_id	gauge	
node_network_iface_link	gauge	
node_network_iface_link_mode	gauge	

node_network_info	gauge	
node_network_mtu_bytes	gauge	
node_network_name_assign_type	gauge	
node_network_net_dev_group	gauge	
node_network_protocol_type	gauge	
node_network_receive_bytes_total	counter	
node_network_receive_compressed_total	counter	
node_network_receive_drop_total	counter	
node_network_receive_errs_total	counter	
node_network_receive_fifo_total	counter	
node_network_receive_frame_total	counter	
node_network_receive_multicast_total	counter	
node_network_receive_packets_total	counter	
node_network_speed_bytes	gauge	
node_network_transmit_bytes_total	counter	
node_network_transmit_carrier_total	counter	
node_network_transmit_colls_total	counter	
node_network_transmit_compressed_total	counter	
node_network_transmit_drop_total	counter	
node_network_transmit_errs_total	counter	
node_network_transmit_fifo_total	counter	
node_network_transmit_packets_total	counter	
node_network_transmit_queue_length	gauge	
node_network_up	gauge	
node_nf_conntrack_entries	gauge	
node_nf_conntrack_entries_limit	gauge	
node_procs_blocked	gauge	
node_procs_running	gauge	
node_schedstat_running_seconds_total	counter	

node_schedstat_timeslices_total	counter	
node_schedstat_waiting_seconds_total	counter	
node_scrape_collector_duration_seconds	gauge	
node_scrape_collector_success	gauge	
node_sockstat_FRAG6_inuse	gauge	
node_sockstat_FRAG6_memory	gauge	
node_sockstat_FRAG_inuse	gauge	
node_sockstat_FRAG_memory	gauge	
node_sockstat_RAW6_inuse	gauge	
node_sockstat_RAW_inuse	gauge	
node_sockstat_TCP6_inuse	gauge	
node_sockstat_TCP_alloc	gauge	
node_sockstat_TCP_inuse	gauge	
node_sockstat_TCP_mem	gauge	
node_sockstat_TCP_mem_bytes	gauge	
node_sockstat_TCP_orphan	gauge	
node_sockstat_TCP_tw	gauge	
node_sockstat_UDP6_inuse	gauge	
node_sockstat_UDPLITE6_inuse	gauge	
node_sockstat_UDPLITE_inuse	gauge	
node_sockstat_UDP_inuse	gauge	
node_sockstat_UDP_mem	gauge	
node_sockstat_UDP_mem_bytes	gauge	
node_sockstat_sockets_used	gauge	
node_softnet_dropped_total	counter	
node_softnet_processed_total	counter	
node_softnet_times_squeezed_total	counter	
node_textfile_scrape_error	gauge	

node_time_seconds	gauge	
node_timex_estimated_error_seconds	gauge	
node_timex_frequency_adjustment_ratio	gauge	
node_timex_loop_time_constant	gauge	
node_timex_maxerror_seconds	gauge	
node_timex_offset_seconds	gauge	
node_timex_pps_calibration_total	counter	
node_timex_pps_error_total	counter	
node_timex_pps_frequency_hertz	gauge	
node_timex_pps_jitter_seconds	gauge	
node_timex_pps_jitter_total	counter	
node_timex_pps_shift_seconds	gauge	
node_timex_pps_stability_exceeded_total	counter	
node_timex_pps_stability_hertz	gauge	
node_timex_status	gauge	
node_timex_sync_status	gauge	
node_timex_tai_offset_seconds	gauge	
node_timex_tick_seconds	gauge	
node_udp_queues	gauge	
node_uname_info	gauge	
node_vmstat_pgfault	untyped	
node_vmstat_pgmajfault	untyped	
node_vmstat_pgpgin	untyped	
node_vmstat_pgpgout	untyped	
node_vmstat_pswpin	untyped	
node_vmstat_pswpout	untyped	

- Windows 主机

key	值类型	说明

windows_cpu_clock_interrupts_total	counter	
windows_cpu_core_frequency_mhz	gauge	
windows_cpu_cstate_seconds_total	counter	
windows_cpu_dpcs_total	counter	
windows_cpu_idle_break_events_total	counter	
windows_cpu_interrupts_total	counter	
windows_cpu_parking_status	gauge	
windows_cpu_processor_performance	gauge	
windows_cpu_time_total	counter	
windows_cs_hostname	gauge	
windows_cs_logical_processors	gauge	
windows_cs_physical_memory_bytes	gauge	
windows_exporter_build_info	gauge	
windows_exporter_collector_duration_seconds	gauge	
windows_exporter_collector_success	gauge	
windows_exporter_collector_timeout	gauge	
windows_exporter_perflib_snapshot_duration_seconds	gauge	
windows_logical_disk_free_bytes	gauge	
windows_logical_disk_idle_seconds_total	counter	
windows_logical_disk_read_bytes_total	counter	
windows_logical_disk_read_latency_seconds_total	counter	
windows_logical_disk_read_seconds_total	counter	
windows_logical_disk_read_write_latency_seconds_total	counter	
windows_logical_disk_reads_total	counter	
windows_logical_disk_requests_queued	gauge	
windows_logical_disk_size_bytes	gauge	
windows_logical_disk_split_ios_total	counter	
windows_logical_disk_write_bytes_total	counter	
windows_logical_disk_write_latency_seconds_total	counter	
windows_logical_disk_write_seconds_total	counter	
windows_logical_disk_writes_total	counter	
windows_net_bytes_received_total	counter	
windows_net_bytes_sent_total	counter	
windows_net_bytes_total	counter	

windows_net_current_bandwidth	gauge	
windows_net_packets_outbound_discarded	counter	
windows_net_packets_outbound_errors	counter	
windows_net_packets_received_discarded	counter	
windows_net_packets_received_errors	counter	
windows_net_packets_received_total	counter	
windows_net_packets_received_unknown	counter	
windows_net_packets_sent_total	counter	
windows_net_packets_total	counter	
windows_os_info	gauge	
windows_os_paging_free_bytes	gauge	
windows_os_paging_limit_bytes	gauge	
windows_os_physical_memory_free_bytes	gauge	
windows_os_process_memory_limix_bytes	gauge	
windows_os_processes	gauge	
windows_os_processes_limit	gauge	
windows_os_time	gauge	
windows_os_timezone	gauge	
windows_os_users	gauge	
windows_os_virtual_memory_bytes	gauge	
windows_os_virtual_memory_free_bytes	gauge	
windows_os_visible_memory_bytes	gauge	
windows_service_info	gauge	
windows_service_start_mode	gauge	
windows_service_state	gauge	
windows_service_status	gauge	
windows_system_context_switches_total	counter	
windows_system_exception_dispatches_total	counter	
windows_system_processor_queue_length	gauge	
windows_system_system_calls_total	counter	
windows_system_system_up_time	gauge	
windows_system_threads	gauge	
windows_textfile_scrape_error	gauge	

- Linux 进程

key	值类型	说明
namedprocess_namegroup_context_switches_total	counter	
namedprocess_namegroup_cpu_seconds_total	counter	
namedprocess_namegroup_major_page_faults_total	counter	
namedprocess_namegroup_memory_bytes	gauge	
namedprocess_namegroup_minor_page_faults_total	counter	
namedprocess_namegroup_num_procs	gauge	
namedprocess_namegroup_num_threads	gauge	
namedprocess_namegroup_oldest_start_time_seconds	gauge	
namedprocess_namegroup_open_filedesc	gauge	
namedprocess_namegroup_read_bytes_total	counter	
namedprocess_namegroup_states	gauge	
namedprocess_namegroup_thread_context_switches_total	counter	
namedprocess_namegroup_thread_count	gauge	
namedprocess_namegroup_thread_cpu_seconds_total	counter	
namedprocess_namegroup_thread_io_bytes_total	counter	
namedprocess_namegroup_thread_major_page_faults_total	counter	
namedprocess_namegroup_thread_minor_page_faults_total	counter	
namedprocess_namegroup_threads_wchan	gauge	
namedprocess_namegroup_worst_fd_ratio	gauge	
namedprocess_namegroup_write_bytes_total	counter	
namedprocess_scrape_errors	counter	
namedprocess_scrape_partial_errors	counter	
namedprocess_scrape_procread_errors	counter	

附录六：告警规则

> Awesome Prometheus alerts

附录七：故障等级划分

故障等级按照《信息系统安全等级保护基本要求》具体划分为四个等级，一级和二级故障为重大故障；三级和四级故障为一般性故障。

一级故障

系统发生故障，预计将已经严重影响公司生产业务系统，导致相关生产业务系统中断1小时以上，并预计24小时以内无法恢复的，具备以下一个或几个特征，既定义为一级故障。

1. 公司机房网络与公有云 VPC 网络出现故障，导致工作人员和用户无法访问相关业务系统；
2. WEB 网站和 APP 系统等关键服务器宕机或有其他原因导致拒绝提供服务的；
3. 利用技术手段造成业务数据被修改、假冒、泄漏、窃取的信息系统安全事件；
4. 由病毒造成关键业务系统不能正常提供服务。

二级故障

信息系统发生故障，预计将或已经严重影响公司生产业务系统，导致相关生产业务系统中断1小时以上，并预计24小时以内可以恢复的，具备以下一个或几个特征，即定义为二级故障。

1. 公司机房网络与公有云 VPC 出现线路和设备故障；
2. WEB 网站和 APP 系统等关键服务器宕机或有其他原因导致拒绝提供服务的；
3. 12 小时以内无法解决的三级故障。

三级故障

满足以下条件之一，即定义为三级故障。

1. 故障发生后，影响到信息系统的运行效率，速度变慢，但不影响业务系统访问；
2. 故障发生后预计在12小时以内恢复；
3. 24小时以内无法解决的四级故障

四级故障

满足以下条件之一，即定义为四级故障。

1. 故障发生后，可随时应急处理，不会影响系统的全面运行；
2. 生产业务系统设备因病毒攻击等原因，造成网络数据出现偶尔掉包，但不影响系统的正常访问和运行。

参考

- 标准
  - **GB/T 37938-2019 信息技术 云资源监控指标体系**
  - GB/T 37736-2019 信息技术 云计算 云资源监控通用要求
  - GB/T 22239-2019 信息安全技术 网络安全等级保护基本要求
- 从自动化到智能化，58 智能监控运维实践
- IBM-数据化运维管理， 让你更为从容地面对IT运维的挑战
- 饿了么监控体系：从架构的减法中演进而来
- 腾讯运维总监聂鑫：海量监控体系是如何炼成的？
- 方法论与技术栈双管齐下的运维可用性能力建设
- 监控之美——Prometheus云原生监控
- 运维，关于监控的那些事，你有必要了解一下
- 运维必知必会的监控知识体系全梳理
- 快看！一张思维导图，包罗最全监控体系建设要点
- 聊透监控体系，就这一篇够不够？

- Metrics, tracing 和 logging 的关系
- 【干货】下一代监控运维体系概述