DAMS

中国数据智能管理峰会

DATA & AI MANAGEMENT SUMMIT

苏宁大规模智能告警收敛与告警根因的实践

汤泳

苏宁科技集团智能监控与运维产研中心总监

SUNING 苏宁

研究方向

大规模时间序列预测、异常检测及因果发现

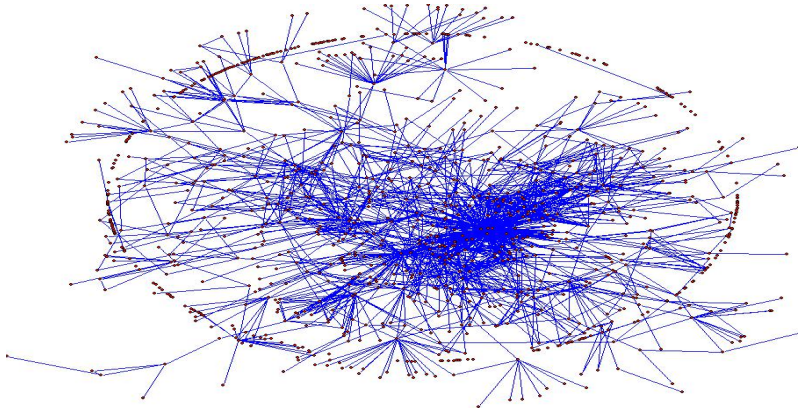
致力于利用深度学习技术对海量时间序列进行建模预测分析，结合真实和预测数据做异常检测。以AI取代缓慢易错的人力决策部分，快速发现问题并且给出决策建议或提前规避故障。

知识图谱构建

从事知识图谱的相关技术研究，致力于利用知识图谱对真实业务场景中大量的告警事件进行告警收敛和根因定位。

目录

基础设施复杂性



系统和服务的复杂性:

- 6000+系统，数量还在增加
- 系统间调用方式复杂：大部分使用RSF，也有HTTP、HESSIAN等
- 苏宁业务的复杂：既有线上新业务又有线下老业务，这些业务系统之间会有大量的关联

基础环境的复杂性:

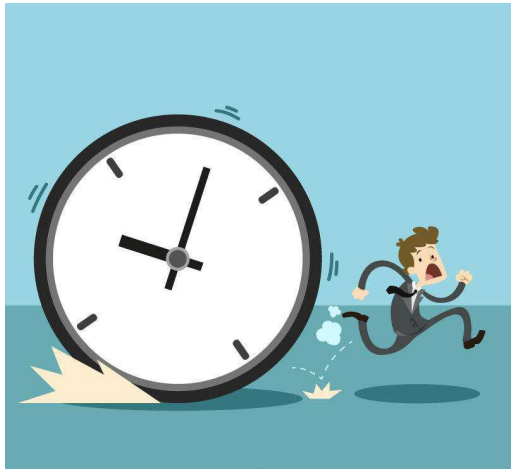
- 多数据中心，每个数据中心会划分多个逻辑机房和部署环境
- 服务器规模30w+，例如，缓存服务器就有可能有上千台服务器
- 设备复杂性：多品牌的交换机，路由器，负载均衡，OpenStack, KVM, k8s下docker，swarm下的docker等

要解决的痛点

告警风暴

定位困难

波及范围广



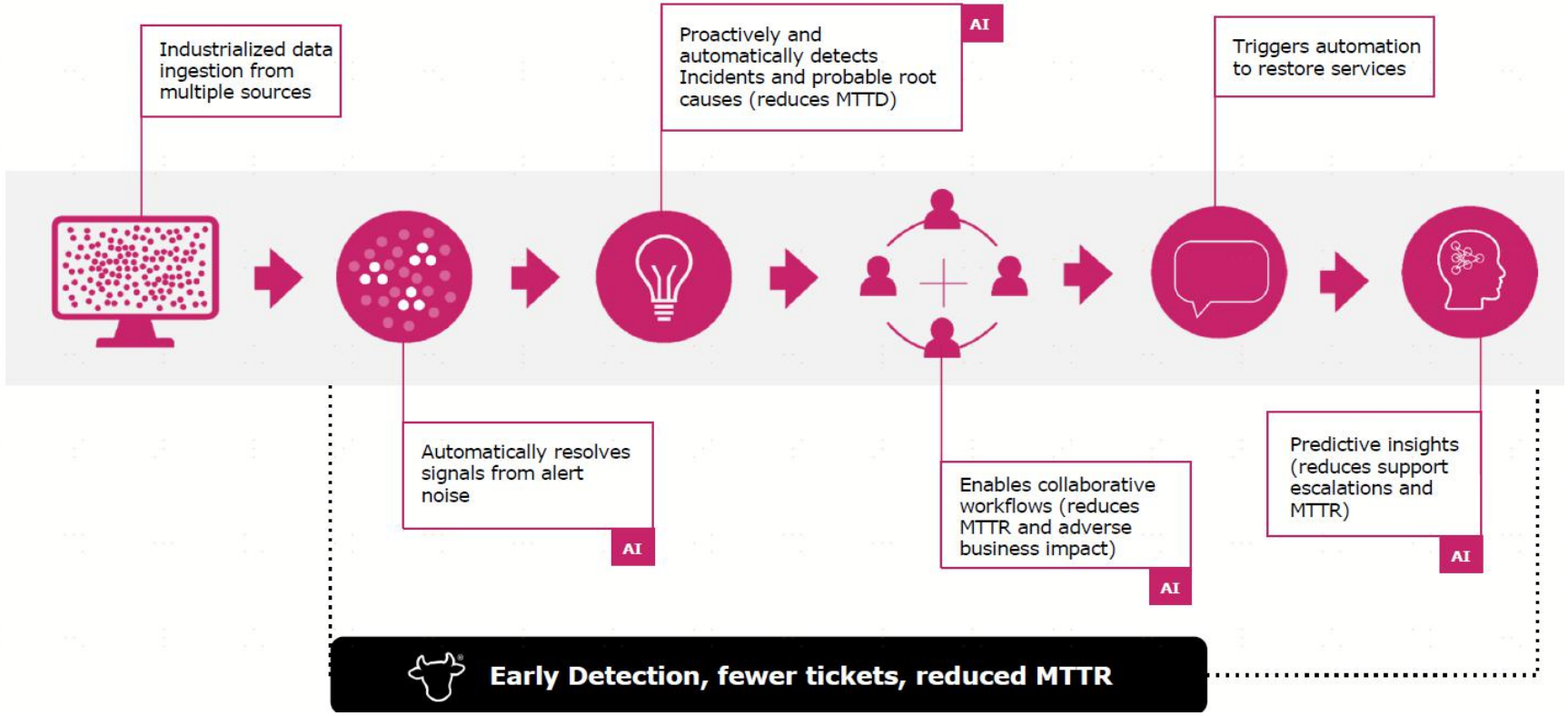
每天产生的告警事件10w+，峰值20w+/天

痛点：

- 面对海量运维监控数据，系统和指标间关联关系越来越复杂，一个节点出现故障，极易引发告警风暴，波及更广的范围，导致定位问题费时费力。
- 此时，单纯依靠人肉和经验分析，越来越难以为继。
- 迫切需要一个工具，可以辅助我们分析系统和指标间关联关系，可视化展示告警的传播路径和影响范围等。

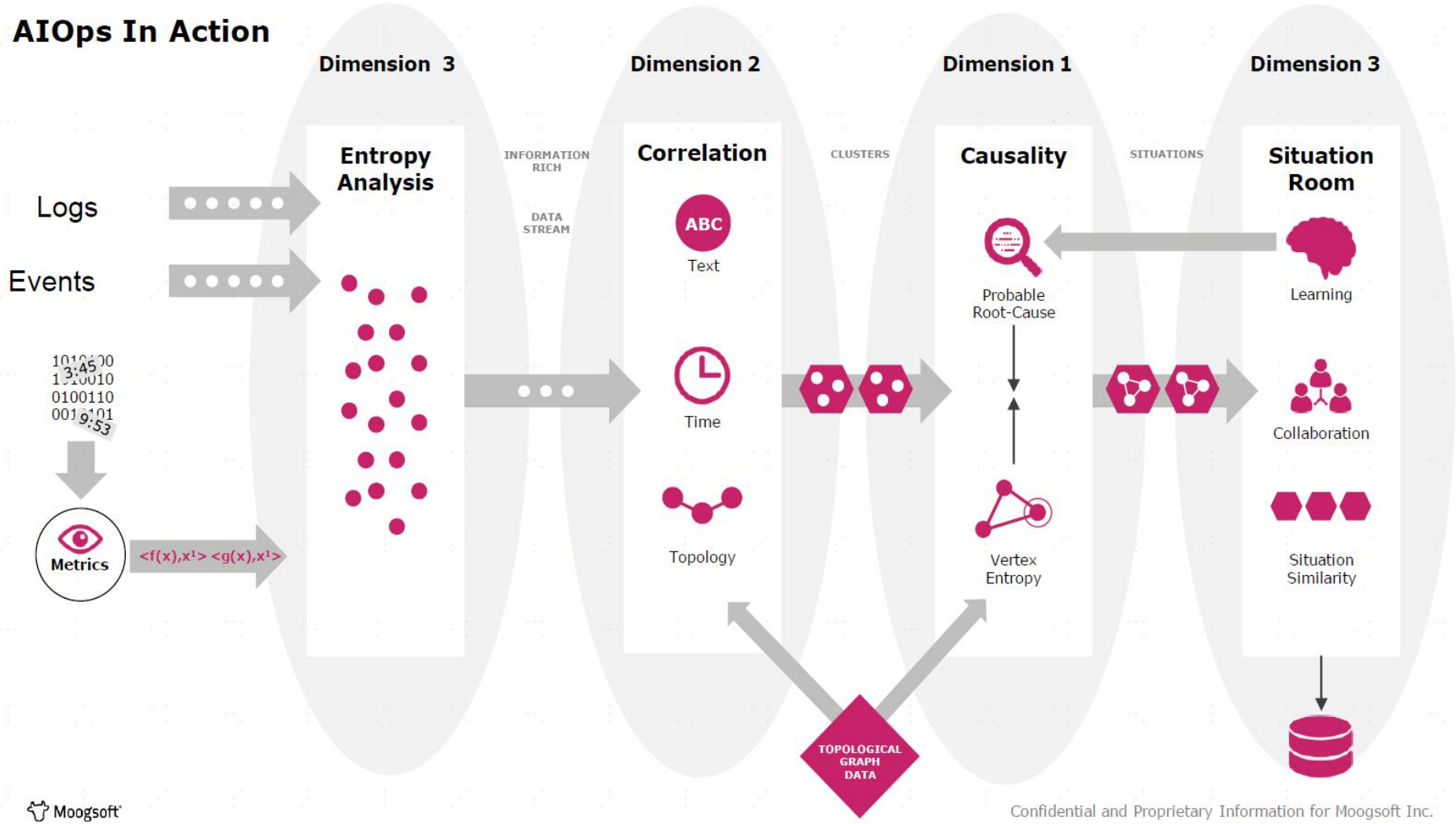
对策

针对上述痛点，我们采用领域知识结合AI的方法对告警进行收敛，以缓解告警风暴。此外，为便于一线运维人员快速的作出干预决策，我们同时对告警的传播路径和影响范围进行分析。



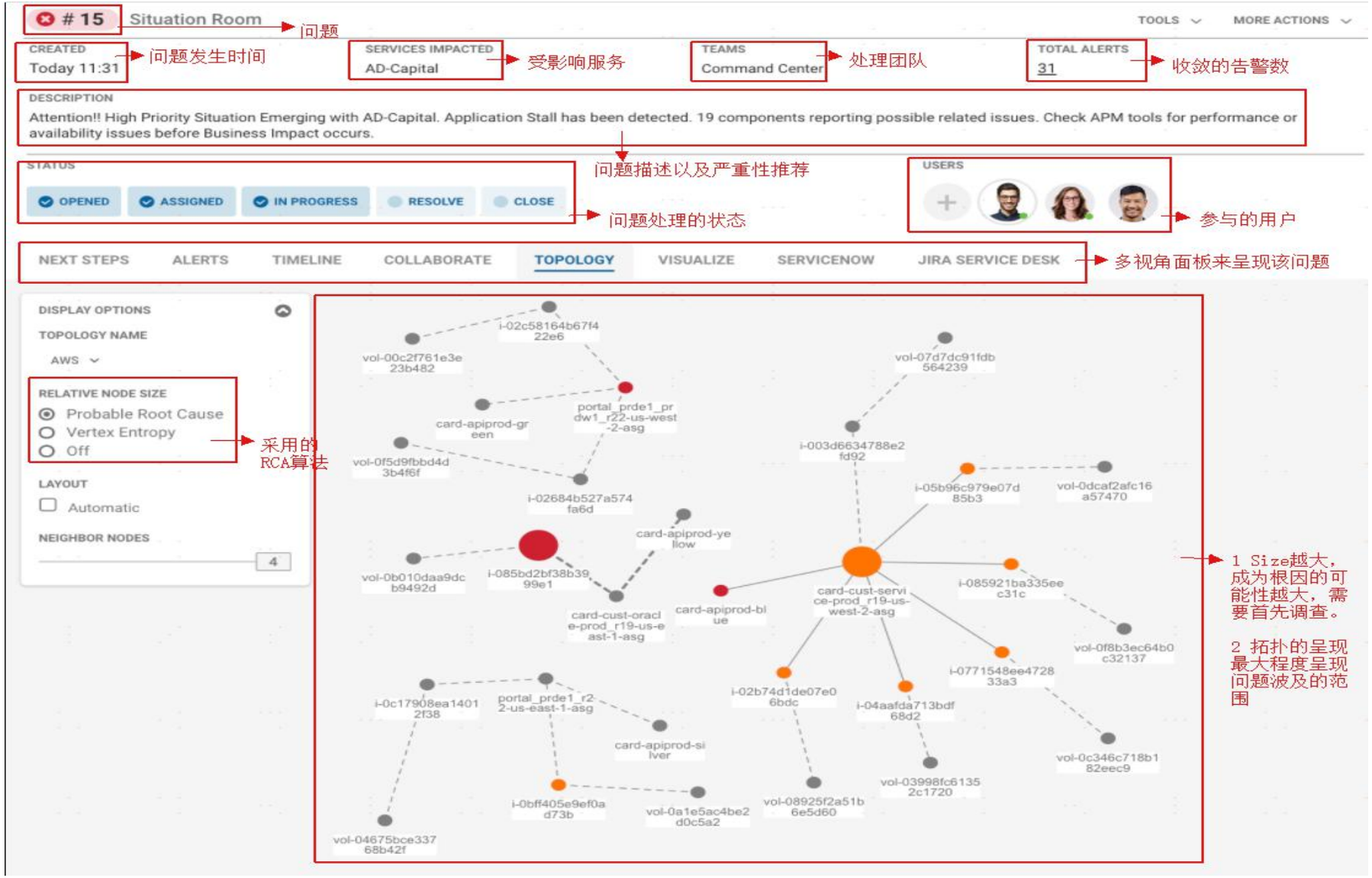
https://www.slideshare.net/Moogsoft/webinar-slides-how-keybank-liberated-its-it-ops-from-rulesbased-event-management

工业级领先的案例：Moogsoft AIOps



https://www.moogsoft.com/

Moogsoft智能告警的产品形态: Situation Room



https://docs.moogsoft.com/AIOps.7.1.0/Situation_Rooms.html

目录

演进路线(1.0)

1.0 版本	2.0 版本	3.0 版本
基于交叉熵的告警聚类	基于GRANO告警根因定位	基于运维知识图谱的告警收敛和根因定位

1. 基于交叉熵的告警聚类

按照告警的场景和规则，利用交叉熵对告警信息进行聚类，实现告警的收敛。借鉴moogsoft的思想，将告警聚类结果生成situation，同一个situation中包含同场景、有关联的告警。

缺点：

- 收敛效果有限，该方法只能减少30%左右的告警，无法有效解决告警风暴问题；
- 无法提供根告警以及根因链路
- 弱解释性
- 无法解决告警的根因问题。

演进路线(2.0)

2. 基于GRANO[2]算法的根因定位

根因定位是在告警收敛的基础上进行的，采用GRANO[2]算法，基于告警收敛结果生成situation，计算situation中每个告警节点的得分，然后排序来确定根因。

缺点：

- 这种方法的缺点是不会给出一条完整的根因链路，因此根因的可解释性不强。

收敛和根因定位结果样例：

演进路线(3.0)

3. 基于运维知识图谱的告警收敛和根因定位

包括全局视角下的软硬件知识图谱和告警知识图谱，利用NLP技术对告警文本信息进行分类，然后将告警收敛到软硬件知识图谱的相关节点上，再结合具有因果关系的告警知识图谱，得出一条 “A –> B –> C –> D” 的根因链路。

优点：

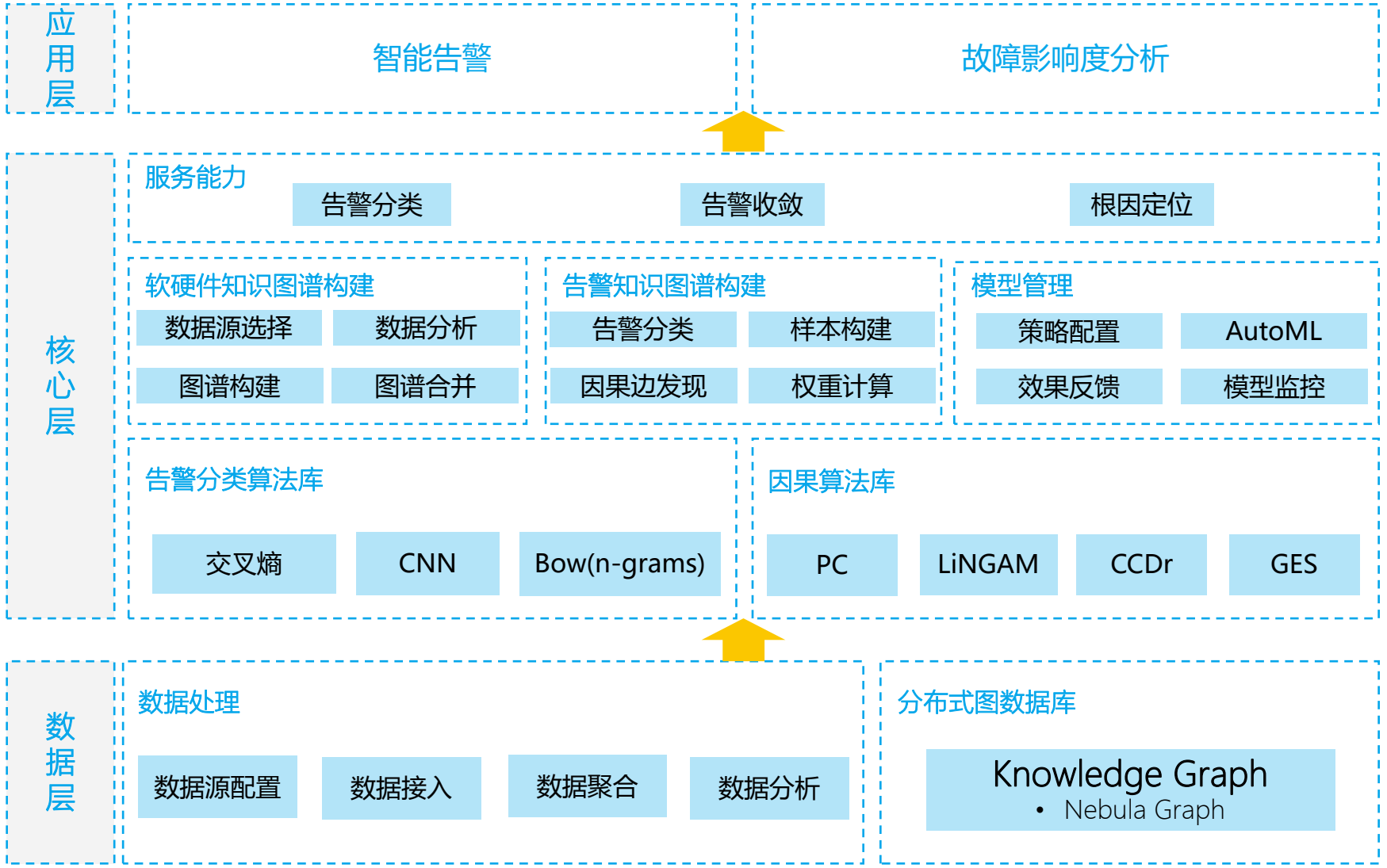
- 由于结合了领域相关知识，该方法收敛效果更好，而且提供了一条完整的根因链路，所以解释性更强，可以更好的为SRE/运维人员提供指引。

收敛和根因定位结果样例

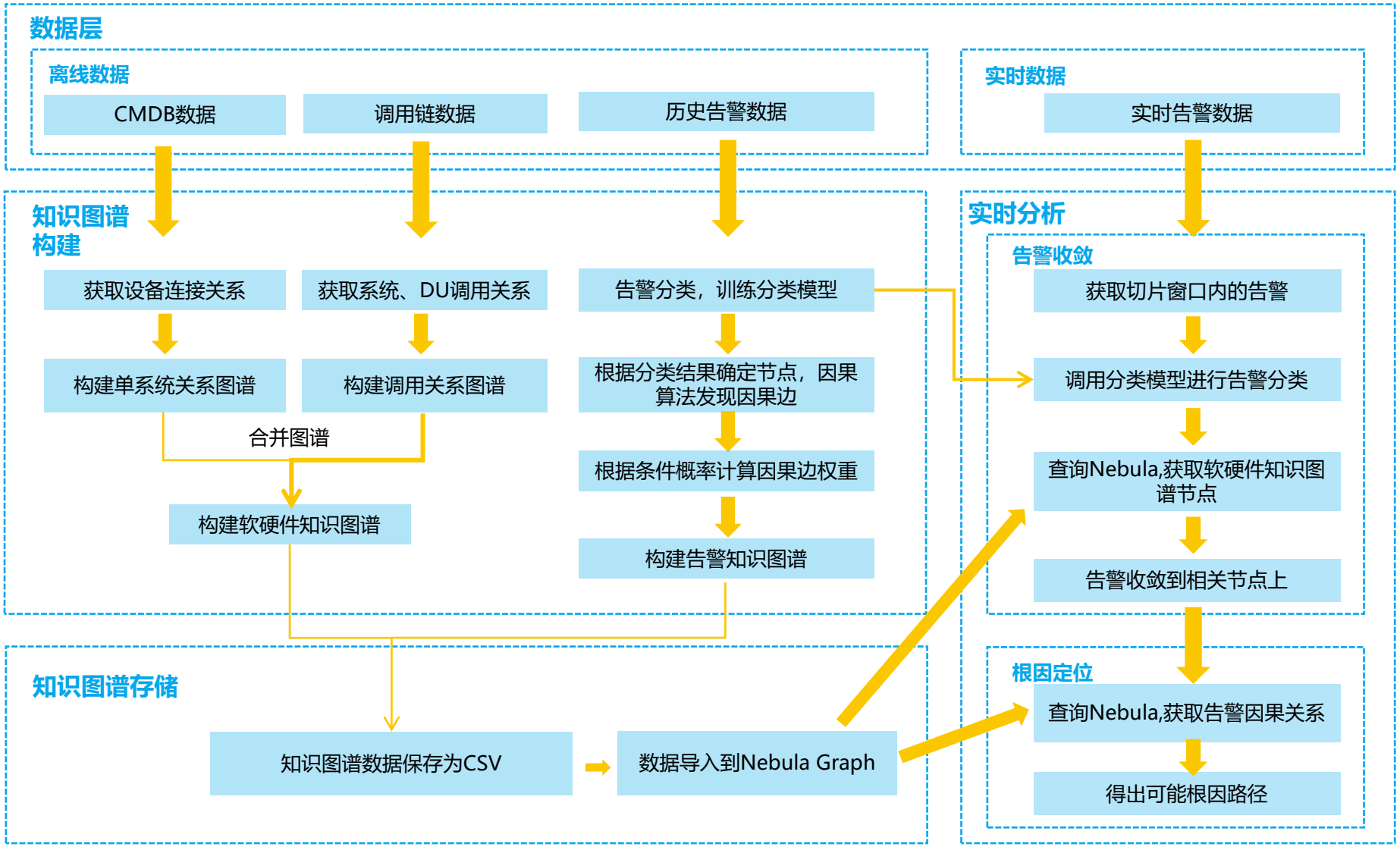
```
收敛结果
{'[redacted]系统(CSIFRLS)':
    {'VM_[redacted]': ['vm-服务器宕机', 'vm-磁盘IO使用率高'],
     'HOST_[redacted]': ['host-磁盘IO使用率高']}}

根因定位
['[redacted]-host-磁盘IO使用率高','[redacted]-vm-磁盘IO使用率高', '[redacted]-vm-服务器宕机']
```

目录

分层建设思路


应用层
智能告警
故障影响度分析
核心层
服务能力
告警分类
告警收敛
根因定位
软硬件知识图谱构建
数据源选择
数据分析
图谱构建
图谱合并
告警知识图谱构建
告警分类
样本构建
因果边发现
权重计算
模型管理
策略配置
AutoML
效果反馈
模型监控
告警分类算法库
交叉熵
CNN
Bow(n-grams)
因果算法库
PC
LiNGAM
CCDr
GES
数据层
数据处理
数据源配置
数据接入
数据聚合
数据分析
分布式图数据库
Knowledge Graph
• Nebula Graph

流程/架构


数据层
离线数据
CMDB数据
调用链数据
历史告警数据
实时数据
实时告警数据
知识图谱构建
获取设备连接关系
获取系统、DU调用关系
告警分类，训练分类模型
构建单系统关系图谱
构建调用关系图谱
根据分类结果确定节点，因果算法发现因果边
合并图谱
根据条件概率计算因果边权重
构建软硬件知识图谱
构建告警知识图谱
实时分析
告警收敛
获取切片窗口内的告警
调用分类模型进行告警分类
查询Nebula,获取软硬件知识图谱节点
告警收敛到相关节点上
根因定位
查询Nebula,获取告警因果关系
得出可能根因路径
知识图谱存储
知识图谱数据保存为CSV
数据导入到Nebula Graph

目录

4.1 引入图数据库

图数据库是以图数据结构存储和查询数据，图包含节点和边。构建运维知识图谱做根告警分析等场景时，为了实时查询分析数据的关联关系，我们将知识图谱持久化存储到图数据库中。另外，引入图数据库还有以下**优势**：

- 图数据库在处理关联数据时的速度快，而关系型数据库在处理反向查询以及多层次关系查询的时候通常开销较大，无法满足我们的要求。
- 图数据库可解释性好。图数据库基于节点和边，以一种直观的方式表示这些关系， 具有天然可解释性。
- 图数据库查询语句表达性好，比如查询一跳，两跳数据，不需要像关系型数据库那样做复杂的表关联。
- 图数据库更灵活。图这种通用结构可以对各种场景进行建模，如社交网络、道路系统等。不管有什么新的数据需要存储，都只需要考虑节点属性和边属性。不像关系型数据库中，需要更新表结构，还要考虑和其他表的关联关系等。

4.1.1图数据库选型

开源图数据库	API	存储引擎	分布式	语言	类型
Neo4j	Cypher	内置	x	Java	图数据库
ArangoDB	类SQL	内置RocksDB	√	C++	多模态数据库
Nebula Graph	类SQL	内置RocksDB	√	C++	图数据库

- Neo4j开源版本不支持分布式，无法满足我们对多副本的需求；
- ArangoDB是多模态数据库，支持graph, document, key/value 和search，支持分布式部署，查询速度快；
- Nebula Graph一款是国产的开源图数据库，支持分布式部署且部署方式比ArangoDB更轻便，查询速度快，腾讯、京东等公司内部也在使用。
- 在充分比较了以上图数据库的性能，以及社区的活跃性以及开放性后，我们最终选择Nebula Graph。

注：我们做了一个详细的性能Benchmark对比

https://discuss.nebula-graph.com.cn/t/topic/1466

4.1.2 Nebula Graph性能测评

测试数据：

- 1632803点
- 30622564边

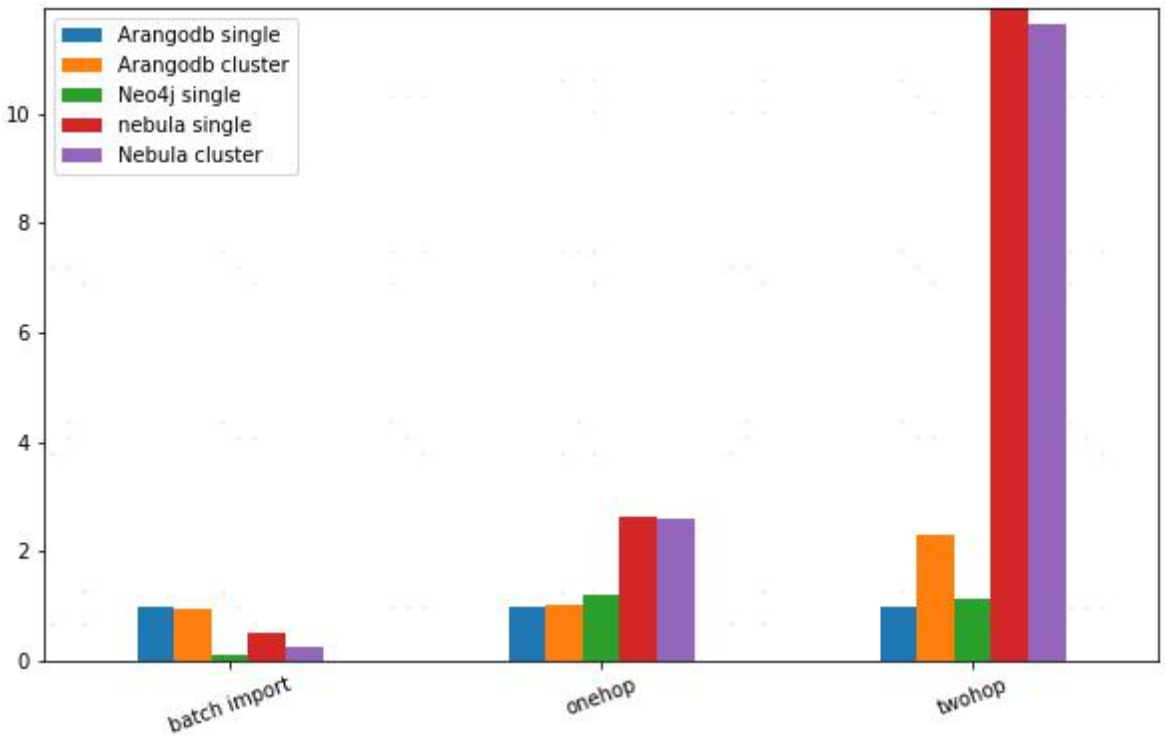
测试机器配置：

- CPU：16c
- 内存：128G
- 磁盘：2400G（HDD）

单机：一台上述机器

集群：三台上述机器

性能对比图



性能测试结果如下：

场景	ArangoDB单机	ArangoDB集群	Neo4j	Nebula单机	Nebula集群
导入时间	292s	283s	32s	150s	71s
onehop	0.34ms	0.35ms	0.41ms	0.89ms	0.87ms
twohop	1.87ms	4.33ms	2.15ms	22.35ms	21.78ms

4.1.3 Nebula扩展能力和容错机制

开源图数据库	扩展能力	容错机制
Neo4j	单机，不支持水平扩展	单机，无容错机制
ArangoDB	支持集群，但数据量大时，插入速度慢。	支持
Nebula	支持集群，可容纳千亿个顶点和万亿条边，并提供毫秒级查询延时	支持

Nebula扩展能力：

Nebula支持集群，可以很方便的进行水平扩展。最多可容纳千亿个顶点和万亿条边，并提供毫秒级查询延时。

Nebula容错机制：

Nebula多个storaged之间，利用RAFT实现数据的多副本强一致。当集群中某台机器出现问题时，能够保证数据的正确性。

4.1.4 Nebula Graph的操作性

查询点的属性：

- FETCH PROP ON kg_node hash("HOST_10.22.125.214") ;

查询边的属性：

- FETCH PROP ON kg_edge hash("HOST_10.22.125.214") -> hash(“VM_10.22.125.213") ;

查询onehop、 twohop范围内的点:

- onehop： GO FROM hash("HOST_10.22.125.214") OVER kg_edge;
- twohop： GO 2 STEPS FROM hash("HOST_10.22.125.214") OVER kg_edge;

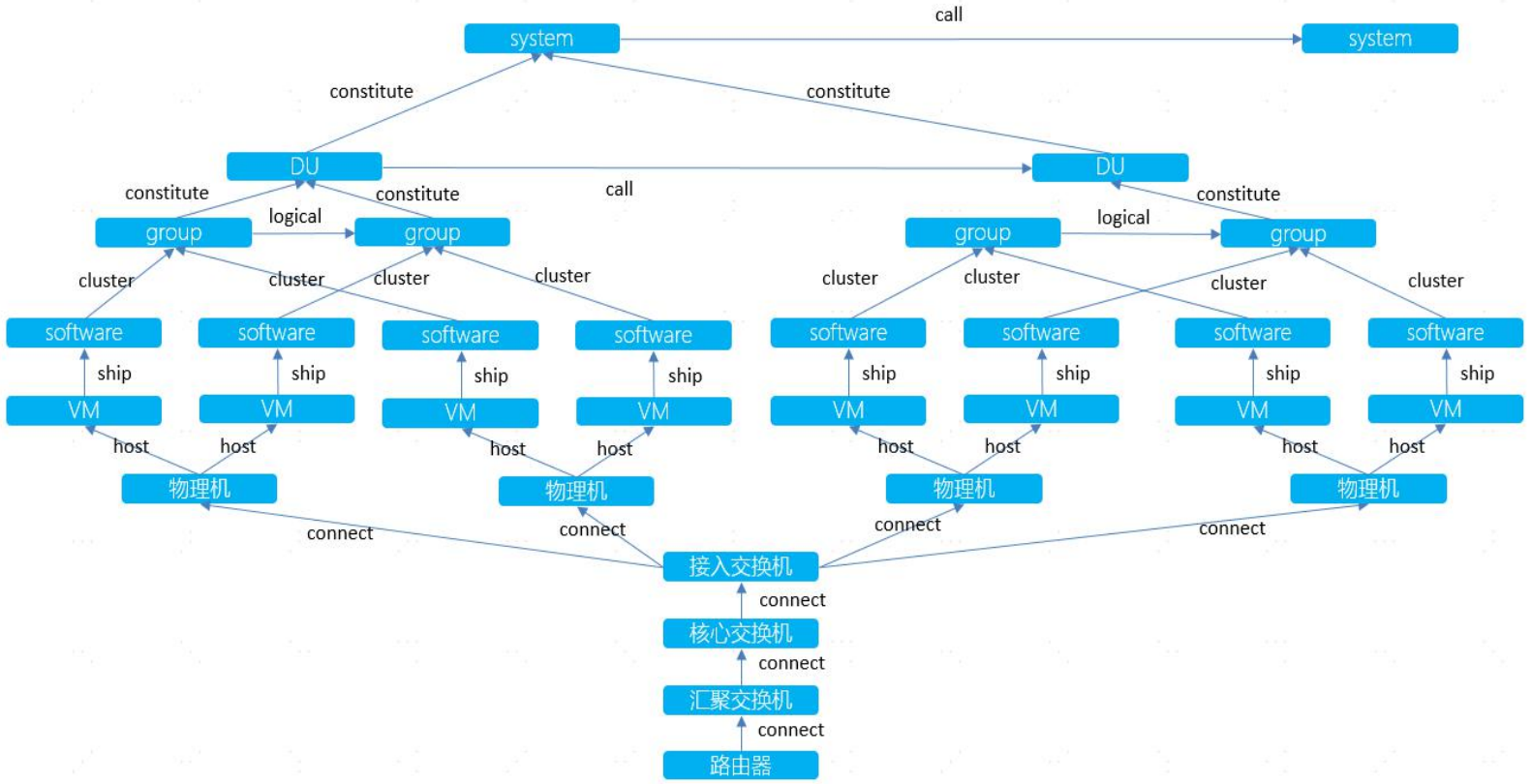
查询子图：我们封装了一个查询子图的方法，只需输入查询的节点列表，即可返回所有节点及其相关关联的边。

- get_subgraph(id_list)

使用参考：https://docs.nebula-graph.io/

4.2 构建软硬件知识图谱

软硬件知识图谱是以全局的视角展示系统内各应用、软件、虚拟机、物理机间的内在逻辑，系统间的调用关系，网络设备的物理连接关系。图谱中的节点包括系统、DU(部署单元)、group(主机实例组)、软件、虚拟机、物理机、接入交换机、核心交换机、汇聚交换机、路由器等。关系包括constitute(构成)、call(调用)、logical(逻辑连接)、cluster(汇聚)、ship(承载)、host(宿主)、connect(物理连接)等，其原型如下：

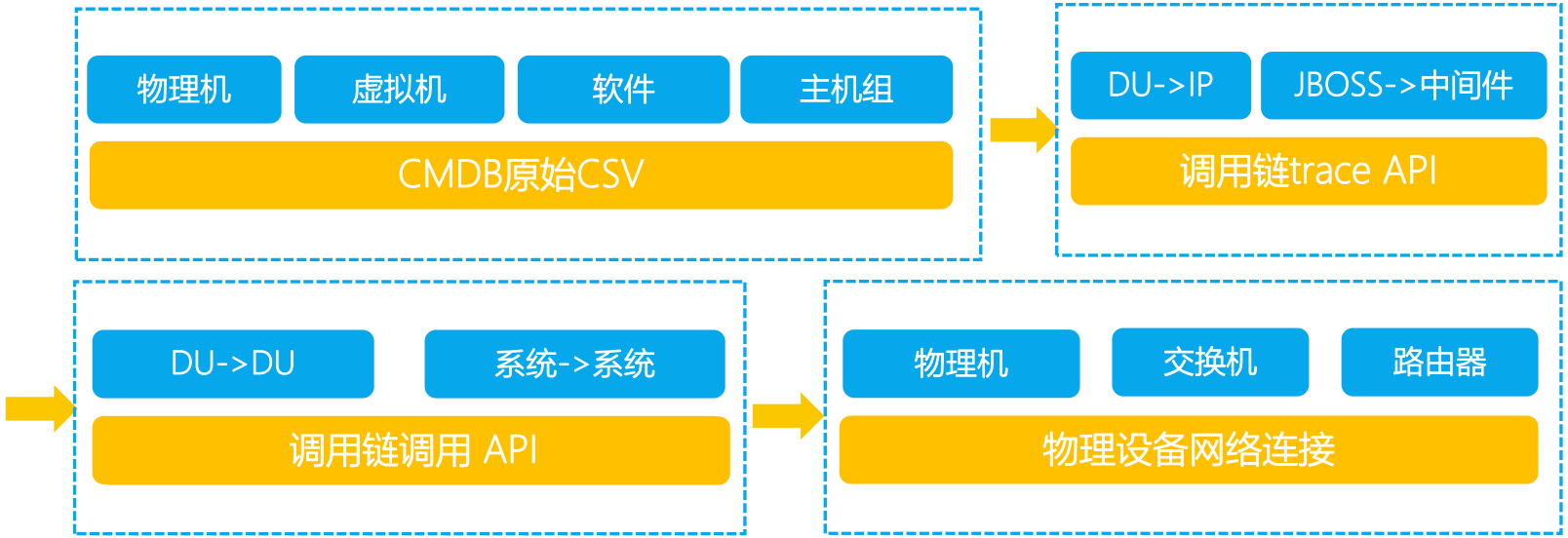


4.2.1 数据源以及构建流程

数据源：

序号	数据源	描述
1	CMDB数据	构建物理机、虚拟机、软件及主机组间的对应关系
2	调用链系统和DU调用关系	构建系统、DU间的调用关系
3	调用链trace详情数据	构建DU->IP映射关系，应用服务器与数据库连接关系
4	物理设备网络连接关系	清晰感知流量路径，链路分析。

构建流程：



4.2.2 CMDB以及链路、设备数据

CMDB数据样例

IP地址	软件类型	软件名称	系统英文名	环境	宿主机IP	操作系统类型	操作系统名称	创建时间	CPU	内存	数据盘	服务实例名称
[redacted]	Other	ESXi 5.0	SNDT	PRD	[redacted]	Linux	VMware ESXi 5.0.0	41938.46875	12	128	4800	(null)
[redacted]	Other	ESXi 5.0	SNDT	PRD	[redacted]	Linux	VMware ESXi 5.0.0	41938.46875	12	128	4800	(null)
[redacted]	Other	ESXi 5.0	SNDT	PRD	[redacted]	Linux	VMware ESXi 5.0.0	41938.46875	12	128	4800	(null)
[redacted]	Other	ESXi 5.0	SNDT	PRD	[redacted]	Linux	VMware ESXi 5.0.0	41938.46875	12	128	4800	(null)
[redacted]	Other	Cloudstac	SNDT	PRD	[redacted]	Linux	CentOS 6.5	41938.46875	12	128	4800	(null)
[redacted]	Other	Cloudstac	SNDT	PRD	[redacted]	Linux	CentOS 6.5	41938.46875	12	128	4800	(null)
[redacted]	Other	Cloudstac	SNDT	PRD	[redacted]	Linux	CentOS 6.5	41938.46875	12	128	4800	(null)
[redacted]	Other	ESXi 5.0	VMWARE	PRD	[redacted]	Linux	VMware ESXi 5.0.0	41938.625	8	24	1800	(null)
[redacted]	Other	ESXi 5.0	VMWARE	PRD	[redacted]	Linux	VMware ESXi 5.0.0	41938.625	8	24	1800	(null)
[redacted]	Other	ESXi 5.0	VMWARE	PRD	[redacted]	Linux	VMware ESXi 5.0.0	41938.625	8	24	1800	(null)

例如：能够构建出

HOST->VM->SOFTWARE->GROUP及GROUP(WildFly)->GROUP(Nginx)的关系图谱

调用链/物理设备数据

调用链数据主要用于获取DU间调用关系、系统间调用关系、DU/IP映射关系、中间件间的逻辑连接关系等。

物理设备主要包括物理机、交换机、路由器等。

4.2.3 合并图谱

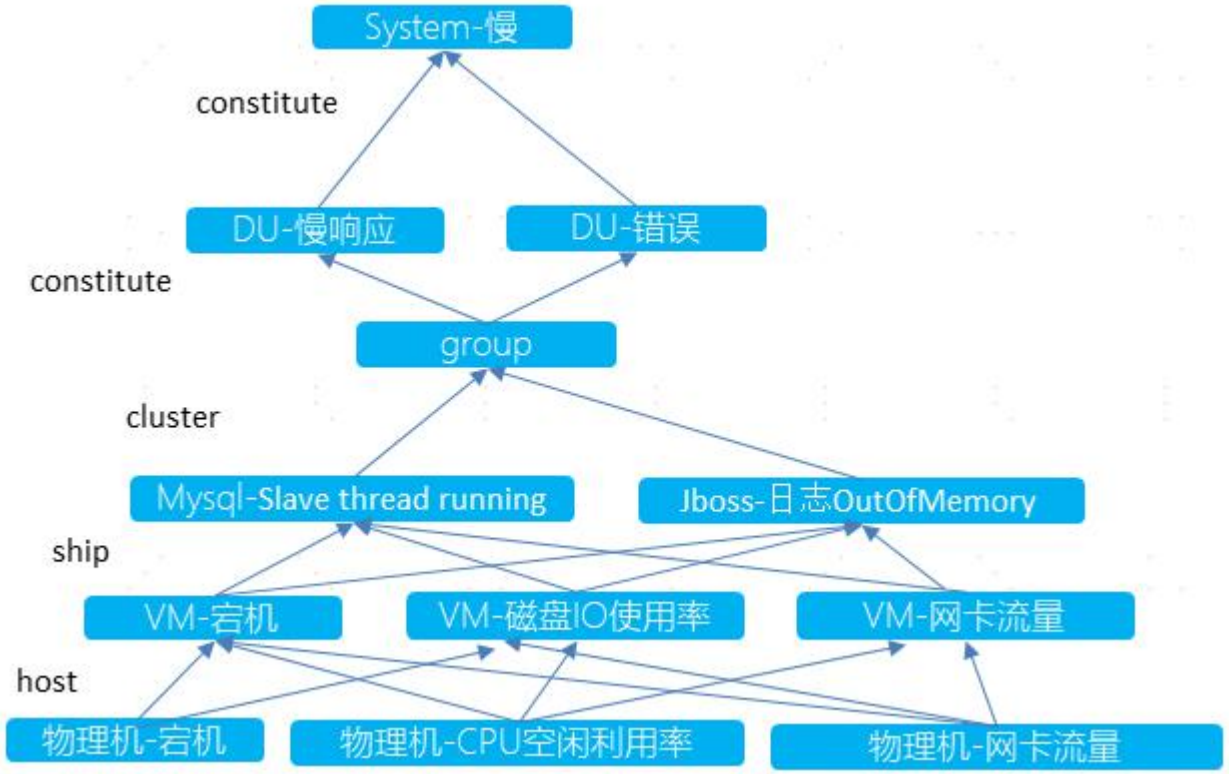
将前面得到的CMDB、调用链和物理设备图谱通过networkx合并，然后存入图数据库中，最终得到的单系统和系统间图谱如下(Nebula Studio可视化呈现)：

其中蓝色节点为系统，浅蓝色节点为DU，绿色节点为group，红色节点为软件，橙色节点为虚拟机，深蓝色节点为物理机，黄色节点为接入交换机，淡黄色节点为汇聚交换机。

Nebula Studio：https://github.com/vesoft-inc/nebula-web-docker

4.3 构建告警知识图谱

告警知识图谱展示的是各种告警之间的因果关系，比如物理机宕机告警会导致虚拟机宕机告警，虚拟机磁盘IO使用率告警会导致机器上一些软件的告警等。关系同样包括constitute(构成)、cluster(汇聚)、ship(承载)、host(宿主) 等，其原型如下：



4.3.1 数据源以及构建流程

数据源：

序号	数据源	描述
1	2019年6月-12月告警数据	通过因果发现算法构建因果图及告警知识图谱

构建流程：

样本构建

告警数据分类
将告警数据分为物理机、虚拟机和软件3大类，并根据告警内容细分为对应的子类

告警数据关联汇聚
利用软硬件知识图谱将有关联的告警汇聚为一组

时间片分割
根据不同时间粒度分割得到样本

因果发现

因果发现算法学习因果图
利用CausalDiscoveryToolbox中的CGNN、PC、GES、LiNGAM等算法对不同时间粒度的样本构建因果边。

告警知识图谱边权重计算
利用告警时序数据及同济大学论文[1]中的方法计算告警知识图谱中边的权重

因果图分析及告警知识图谱构建
对算法构建的因果图进行分析整合，构建告警知识图谱

4.3.2 告警数据样例

IP	系统	软件	告警信息	告警时间	监控项名称	告警级别	machine_type
[redacted]	[redacted]系统 (IMS)	WildFly standalone	主机[redacted](VM)网卡eth0发送流量持续10分钟平均值大于180M，...	2019-05-31 09:49:53	eth0 网卡发送数据流量	警告	vm
[redacted]	[redacted]系统 (IMS)	WildFly standalone	主机[redacted](VM)网卡eth0发送流量持续10分钟平均值大于180M，...	2019-05-31 09:50:47	eth0 网卡发送数据流量	警告	vm
[redacted]	[redacted]平台 (ITSM)	WildFly standalone	主机[redacted]日志中有ITSM错误信息	2019-05-31 09:50:50	ITSM错误日志监控	警告	vm
[redacted]	[redacted]系统 (IMS)	WildFly standalone	主机[redacted](VM)网卡eth0发送流量持续10分钟平均值大于180M，...	2019-05-31 09:51:06	eth0 网卡发送数据流量	警告	vm
[redacted]	[redacted]平台 (RDRS)	WildFly standalone	主机[redacted](VM)网卡eth0发送流量持续10分钟平均值大于400M，当前...	2019-05-31 09:51:50	eth0 网卡发送数据流量	警告	vm

给定时间切片，寻找该时间切片内在软硬件知识图谱上有关联的告警，该组关联的告警为一个因果样本，得到最终的样本集如下，每一行记录为一个因果样本，每条样本中发生告警的列记为1，未发生告警的列记为0。

vm-ping延时过大	vm-端口连接数过大	software-Elastic Search Cluster Status	vm-空闲交换空间少	vm-端口通信异常	vm-空闲CPU少	software-zookeeper zk_max_latency值过大	vm-网卡流量大	software-mysql日志产生Sort aborted信息	vm-虚拟CPU等待实际CPU时间过长	...	software-jboss日志中有OutOfMemoryError	vm-磁盘分区剩余空间少	host-服务器宕机	host-磁盘IO使用率高	vm-CPU利用率高	vm-磁盘IO使用率高	software-mysql Slave_SQL_Running同步进程
0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0

告警类型

4.3.3 因果发现算法

采用已有的因果发现算法工具包：CausalDiscoveryToolbox，其中包含的算法有：PC、GES、CCDr、LiNGAM等。

PC：是因果发现中最著名的基于分数的方法， 该算法对变量和变量集的进行条件测试，以获得可能的因果边。

……

基于构建的因果样本，举例使用PC算法进行因果边发现，输出可能因果节点和边，部分结果如右图所示：算法发现了【host-服务器宕机】导致【vm-服务器宕机】的因果边，符合实际场景。

edge
('host-服务器宕机', 'vm-服务器宕机')
('software-openstack服务异常', 'software-网页访问失败')
('host-磁盘IO使用率高', 'vm-磁盘IO使用率高')
('vm-端口通信异常', 'software-网页访问失败')
('host-服务器宕机', 'software-网页访问失败')
('vm-服务器宕机', 'host-服务器宕机')
('vm-磁盘IO使用率高', 'host-磁盘IO使用率高')
('software-网页访问失败', 'software-openstack服务异常')
('vm-空闲交换空间少', 'vm-网卡流量大')
('vm-服务器宕机', 'software-mysql slave延时过大')

4.3.4 因果边权重计算

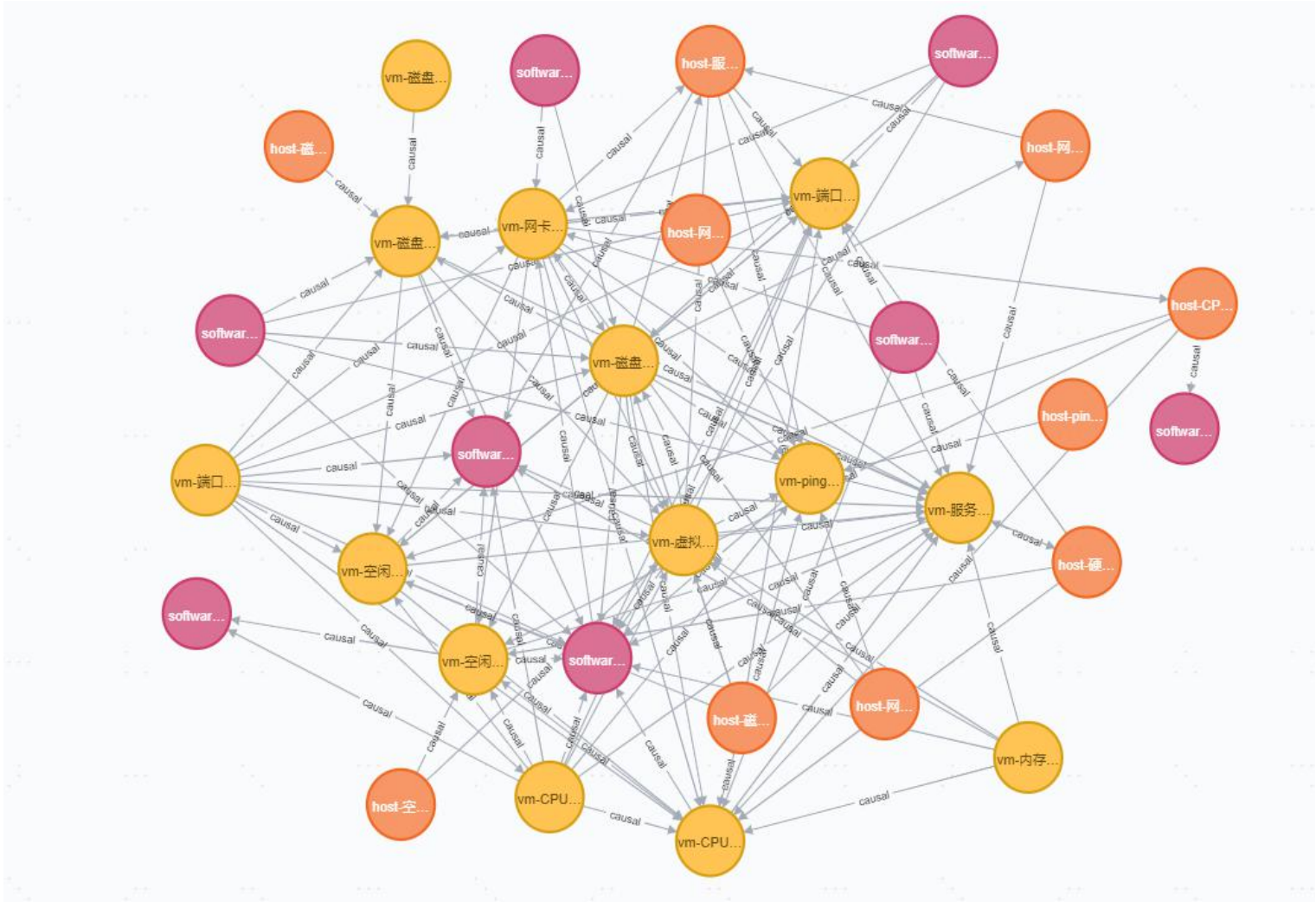
因果边的权重采用条件概率计算，即：基于因果发现样本数据和因果发现算法给出的因果边（包括因果节点），【因节点发生告警的条件下果节点发生告警的次数】与【因节点总共发生的告警次数】的比值作为该因果边的权重。

截取部分的因果边及其权重：【host-服务器宕机】导致【vm-服务器宕机】的因果边权重为0.99，也从侧面验证了因果算法的可行性。

edge	weight
('host-ping延时过大', 'vm-ping延时过大')	1.000000
('host-网卡速率异常', 'vm-服务器宕机')	1.000000
('host-磁盘分区空闲inodes数过少', 'vm-磁盘分区剩余空间少')	1.000000
('host-网卡overruns', 'vm-服务器宕机')	0.991718
('host-服务器宕机', 'vm-服务器宕机')	0.985645
...	...

4.3.4 因果边权重计算

得到的因果关系图存入图数据库，进行可视化展示：



4.4 告警收敛和根因定位

流程：

- 设置起始时间点，获取该时段段内的告警数据；
- 告警分类，根据训练好的分类模型，将告警分为VM、HOST、SOFTWARE及对应的类别；
- 告警收敛：根据告警内容，查询软硬件知识图谱将告警以系统为单位进行收敛，收敛格式如下：

系统1：{软硬件知识图谱节点1：[告警类型1，告警类型2，...]，
软硬件知识图谱节点2：[告警类型1，告警类型2，...]，...}，

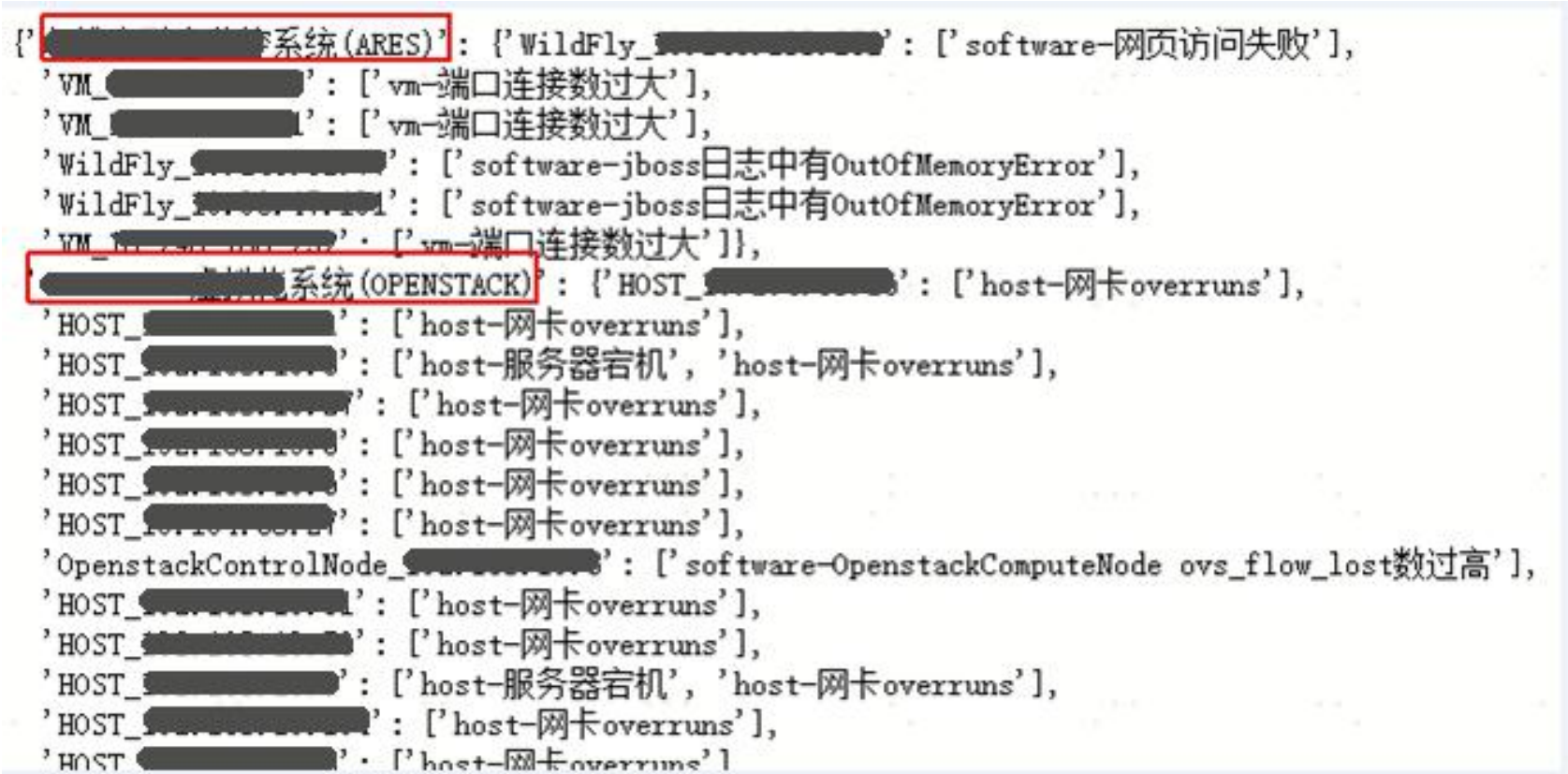
系统2：{软硬件知识图谱节点1：[告警类型1，告警类型2，...]，
软硬件知识图谱节点2：[告警类型1，告警类型2，...]，...}，...};

- 告警因果图构建：根据告警收敛及告警知识图谱，生成告警因果子图。
- 基于生成的告警因果子图及权重计算疑似路径，排序给出根因路径。

4.4.1 告警收敛

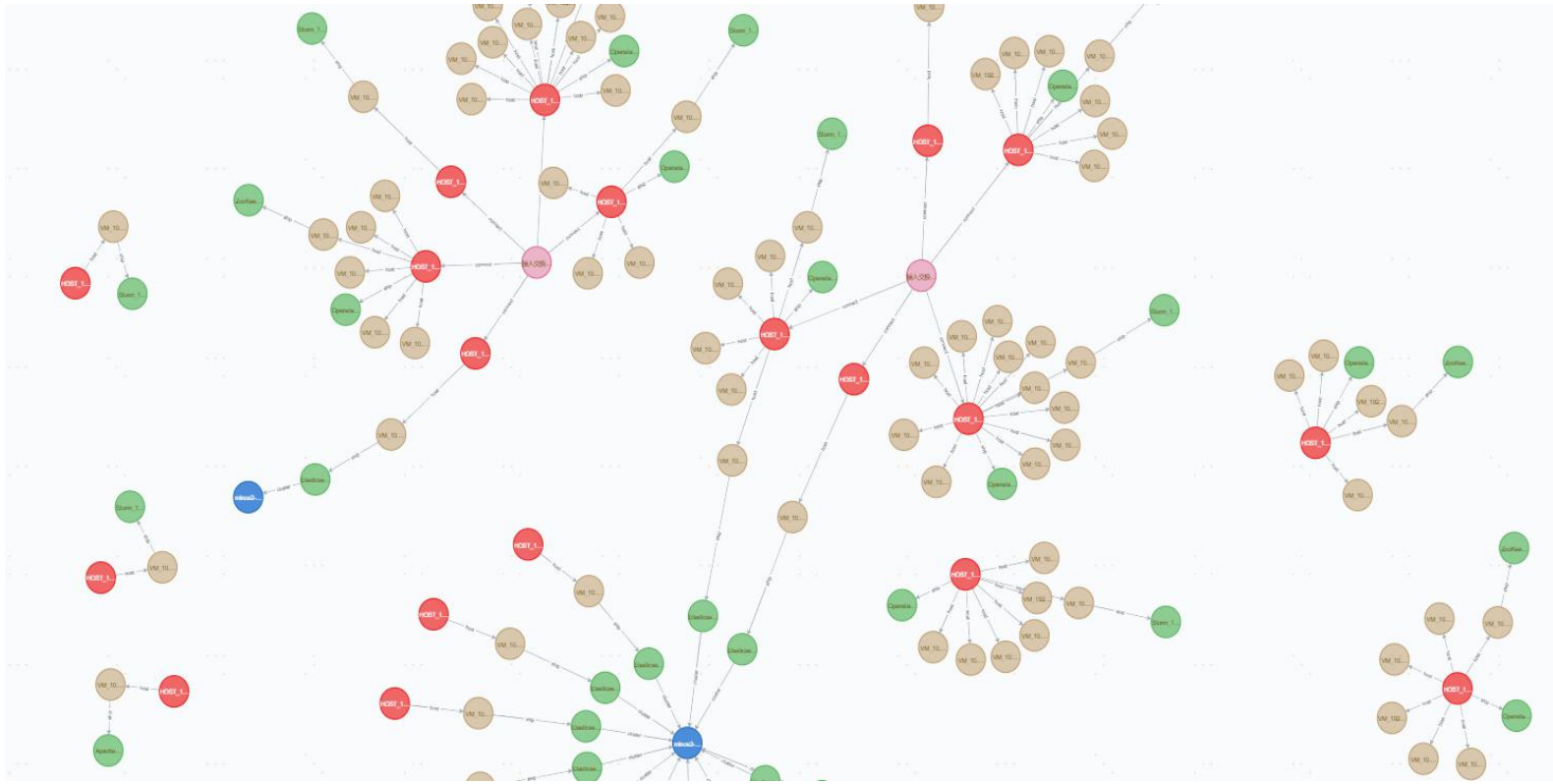
基于已有的告警数据，我们以5min为时间段创建时间切片样本，并取告警数量最多一个时间切片作为主要分析的内容。

根据系统和软硬件知识图谱节点，对该时间段内的告警进行收敛，结果如下：



4.4.2 根因定位

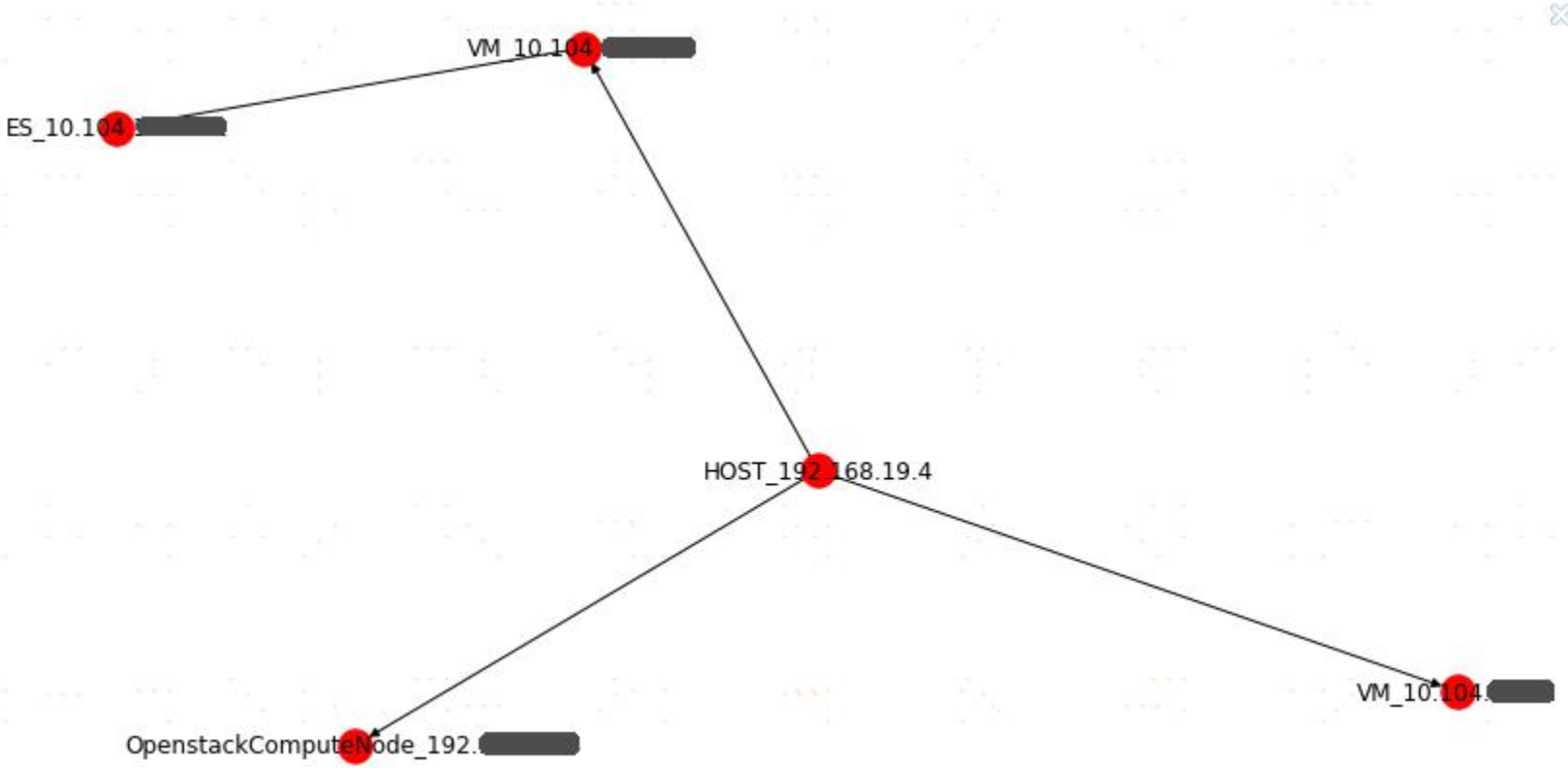
对于告警收敛结果中的某个系统，在图数据库中查询“一跳”范围内该系统下的所有节点构成的子图，以 苏宁的某一个系统为例，该系统下的所有节点如下，（红色表示**物理机**节点，棕色表示**虚拟机**节点，绿色表示**软件**节点）。



—>接上页

上图中出现的所有节点中，既包括有告警信息的，也包括没有告警消息的，因此将上述因果图输入到netwokx后，可以得到最终经过精简后的有告警消息发出的各节点的因果图， 其中一部分的因果图展示如下：

可以解释为：“192.168.xxx.xxx-host-服务器宕机”导致 “10.104.xxx.xxx-vm-服务器宕机”，进而导致“software-网页访问失败”。



进一步的，根据上述生成的因果图，再结合因果图中每条边的权重，就可以计算出该时间切片下的所有**疑似根因路径**，经过排序后即可得到最终的根因路径。本例中最终得到的几条根因路径如下：

可以看出，程序最终给出了几条疑似的根因路径，其中包括最长的一条，可以解释为：**ip为192.168.xxx.xxx 这台物理机 由于网卡overruns 的原因，导致了这台物理机的宕机，从而使得这台物理机上的 ip为10.104.xxx.xxx 的虚拟机宕机，最终导致这台虚拟机上的相关的网页访问失败。**

目录

效果

告警收敛：

经过验证，基于运维知识图谱的告警收敛可缩减至少50%的告警量，最高可达到60%以上，明显优于基于交叉熵的收敛效果，更有效率的缓解了告警风暴的压力；另外，在时效性方面，基于1、2、5min不同长度的时间切片进行告警收敛，耗时可以控制在6s以内，满足告警通知的时效性要求。

根因定位：

经一线运维人员验证，每个告警时间段提供的可能根因传播路径集合基本包含了真实的根因，有效缩短了运维人员的干预时间；另外在耗时上，根因定位可以控制在3s以内，速度较快，满足时效性要求。

改进方向

告警数据源扩展：

当前仅是基于CMDB和Zabbix告警数据构建运维知识图谱进行告警收敛和根因定位，基础设施层面的告警数据更简单、规范，后续还要扩展到更复杂的非基础设施层面的告警数据中，比如业务自定义的告警等。

知识图谱与异常检测的结合：

目前还没有利用知识图谱对异常检测（时间序列数据）结果做根因定位的应用实践，这需要对时间序列做因果关系的发现，构建时间序列之间的因果图，从而打通知识图谱与异常检测的壁垒，这也是知识图谱后续使用的扩展方向之一。

运维知识图谱的提升优化：

告警类型多而复杂，随着告警的增多，会出现更多新的告警类型，这些新的告警类型并不在旧的告警图谱中，所以需要不断的优化告警知识图谱，同时对准确率做进一步提升。

Papers & Action

[1] A Causality Mining and Knowledge Graph Based Method of Root Cause Diagnosis for Performance Anomaly in Cloud Applications.

[2] GRANO: Interactive Graph-based Root Cause Analysis for Cloud-Native Distributed Data Platform.

[3] Moogsoft AIOps Implementer Training.

[4] Causal discovery and inference: concepts and recent methodological advances.

[5] A Causality Mining and Knowledge Graph BasedMethod of Root Cause Diagnosis for PerformanceAnomaly in Cloud Applications.

[6] https://github.com/vesoft-inc/nebula

[7] https://fentechsolutions.github.io/CausalDiscoveryToolbox/html/index.html

DAMS
中国数据智能管理峰会
DATA & AI MANAGEMENT SUMMIT

Q&A

DAMS

中国数据智能管理峰会

DATA & AI MANAGEMENT SUMMIT

THANK YOU!

SUNING 苏宁