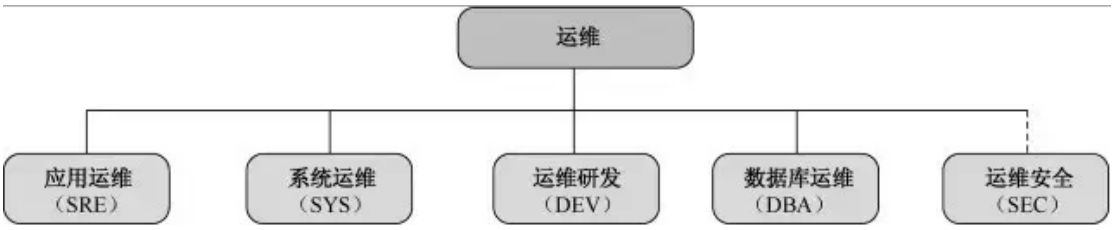
运维工作解释
2017/3/9

PC 小记：《运维之下》的文档作者对运维工作的解释清晰明了，有利于工作的划分和职业的规划，因而值得整理后分享。

一、 运维工作分类

运维的工作方向比较多，随着业务规模的不断发展，越成熟的互联网公司，运维岗位会划分得越细。当前很多大型的互联网公司，在初创时期只有系统运维，随着业务规模、服务质量的要求，也逐渐进行了工作细分。一般情况下运维团队的工作分类（见图 1-1）和职责如下。



系统运维

系统运维负责 IDC、网络、CDN 和基础服务的建设（LVS、NTP、DNS）；负责资产管理，服务器选型、交付和维修。详细的工作职责如下：

（1）IDC 数据中心建设

收集业务需求，预估未来数据中心的发展规模，从骨干网的分布，数据中心建筑，以及 Internet 接入、网络攻击防御能力、扩容能力、空间预留、外接专线能力、现场服务支撑能力等多个方面评估选型数据中心。负责数据中心的建设、现场维护工作。

（2）网络建设

设计及规划生产网络架构，这里面包括：数据中心网络架构、传输网架构、CDN 网络架构等，

以及网络调优等日常运维工作。

（3）LVS 负载均衡和 SNAT 建设

LVS 是整个站点架构中的流量入口，根据网络规模和业务需求，构建负载均衡集群；完成网络与业务服务器的衔接，提供高性能、高可用的负载调度能力，以及统一的网络层防攻击能力；SNAT 集中提供数据中心的公网访问服务，通过集群化部署，保证出网服务的高性能与高可用。

（4）CDN 规划和建设

CDN 工作划分为第三方和自建两部分。建立第三方 CDN 的选型和调度控制；根据业务发展趋势，规划 CDN 新节点建设布局；完善 CDN 业务及监控，保障 CDN 系统稳定、高效运行；分析业务加速频道的文件特性和数量，制定最优的加速策略和资源匹配；负责用户劫持等 CDN 日常故障排查工作。

（5）服务器选型、交付和维护

负责服务器的测试选型，包含服务器整机、部件的基础性测试和业务测试，降低整机功率，提升机架部署密度等。结合对公司业务的了解，推广新硬件、新方案减少业务的服务器投入规模。负责服务器硬件故障的诊断定位，服务器硬件监控、健康检查工具的开发和维护。

（6）OS、内核选型和 OS 相关维护工作

负责整体平台的 OS 选型、定制和内核优化，以及 Patch 的更新和内部版本发布；建立基础的 YUM 包管理和分发中心，提供常用包版本库；跟进日常各类 OS 相关故障；针对不同的业务类型，提供定向的优化支持。

（7）资产管理

记录和管理运维相关的基础物理信息，包括数据中心、网络、机柜、服务器、ACL、IP 等各种资源信息，制定有效的流程，确保信息的准确性；开放 API 接口，为自动化运维提供数据支持。

（8）基础服务建设

业务对 DNS、NTP、SYSLOG 等基础服务的依赖非常高，需要设计高可用架构避免单点，提供稳定的基础服务。

应用运维

应用运维负责线上服务的变更、服务状态监控、服务容灾和数据备份等工作，对服务进行例行排查、故障应急处理等工作。详细的工作职责如下所述。

（1）设计评审

在产品研发阶段，参与产品设计评审，从运维的角度提出评审意见，使服务满足运维准入的高可用要求。

（2）服务管理

负责制定线上业务升级变更及回滚方案，并进行变更实施。掌握所负责的服务及服务间关联关系、服务依赖的各种资源。能够发现服务上的缺陷，及时通报并推进解决。制定服务稳定性指标及准入标准，同时不断完善和优化程序和系统的功能、效率，提高运行质量。完善监控内容，提高报警准确度。在线上服务出现故障时，第一时间响应，对已知线上故障能按流程进行通报并按预案执行，未知故障组织相关人员联合排障。

（3）资源管理

对各服务的服务器资产进行管理，梳理服务器资源状况、数据中心分布情况、网络专线及带宽情况，能够合理使用服务器资源，根据不同服务的需求，分配不同配置的服务器，确保服务器资源的充分利用。

（4）例行检查

制定服务例行排查点，并不断完善。根据制定的服务排查点，对服务进行定期检查。对排查过程中发现的问题，及时进行追查，排除可能存在的隐患。

（5）预案管理

确定服务所需的各项监控、系统指标的阈值或临界点，以及出现该情况后的处理预案。建立和更新服务预案文档，并根据日常故障情况不断补充完善，提高预案完备性。能够制定和评

审各类预案，周期性进行预案演练，确保预案的可执行性。

（6）数据备份

制定数据备份策略，按规范进行数据备份工作。保证数据备份的可用性和完整性，定期开展数据恢复性测试。

数据库运维

数据库运维负责数据存储方案设计、数据库表设计、索引设计和 SQL 优化，对数据库进行变更、监控、备份、高可用设计等工作。详细的工作职责如下所述。

（1）设计评审

在产品研发初始阶段，参与设计方案评审，从 DBA 的角度提出数据存储方案、库表设计方案、SQL 开发标准、索引设计方案等，使服务满足数据库使用的高可用、高性能要求。

（2）容量规划

掌握所负责服务的数据库的容量上限，清楚地了解当前瓶颈点，当服务还未到达容量上限时，及时进行优化、分拆或者扩容。

（3）数据备份与灾备

制定数据备份与灾备策略，定期完成数据恢复性测试，保证数据备份的可用性和完整性。

（4）数据库监控

完善数据库存活和性能监控，及时了解数据库运行状态及故障。

（5）数据库安全

建设数据库账号体系，严格控制账号权限与开放范围，降低误操作和数据泄露的风险；加强离线备份数据的管理，降低数据泄露的风险。

（6）数据库高可用和性能优化

对数据库单点风险和故障设计相应的切换方案，降低故障对数据库服务的影响；不断对数据库整体性能进行优化，包括新存储方案引进、硬件优化、文件系统优化、数据库优化、SQL 优化等，在保障成本不增加或者少量增加的情况下，数据库可以支撑更多的业务请求。

（7）自动化系统建设

设计开发数据库自动化运维系统，包括数据库部署、自动扩容、分库分表、权限管理、备份恢复、SQL 审核和上线、故障切换等功能。

运维研发

运维研发负责通用的运维平台设计和研发工作，如：资产管理、监控系统、运维平台、数据权限管理系统等。提供各种 API 供运维或研发人员使用，封装更高层的自动化运维系统。详细的工作职责如下所述。

（1）运维平台

记录和管理服务及其关联关系，协助运维人员自动化、流程化地完成日常运维操作，包括机器管理、重启、改名、初始化、域名管理、流量切换和故障预案实施等。

（2）监控系统

负责监控系统的设计、开发工作，完成公司服务器和各种网络设备的资源指标、线上业务运行指标的收集、告警、存储、分析、展示和数据挖掘等工作，持续提高告警的及时性、准确性和智能性，促进公司服务器资源的合理化调配。

（3）自动化部署系统

参与部署自动化系统的开发，负责自动化部署系统所需要的基础数据和信息，负责权限管理、API 开发、Web 端开发。结合云计算，研发和提供 PaaS 相关高可用平台，进一步提高服务的部署速度和用户体验，提升资源利用率。

运维安全

运维安全负责网络、系统和业务等方面的安全加固工作，进行常规的安全扫描、渗透测试，进行安全工具和系统研发以及安全事件应急处理。详细的工作职责如下所述。

（1）安全制度建立

根据公司内部的具体流程，制定切实可行，且行之有效的安全制度。

（2）安全培训

定期向员工提供具有针对性的安全培训和考核，在全公司内建立安全负责人制度。

（3）风险评估

通过黑白盒测试和检查机制，定期产生对物理网络、服务器、业务应用、用户数据等方面的总体风险评估结果。

（4）安全建设

根据风险评估结果，加固最薄弱的环节，包括设计安全防线、部署安全设备、及时更新补丁、防御病毒、源代码自动扫描和业务产品安全咨询等。为了降低可能泄露数据的价值，通过加密、匿名化、混淆数据，乃至定期删除等技术手段和流程来达到目的。

（5）安全合规

为了满足例如支付牌照等合规性要求，安全团队承担着安全合规的对外接口人工作。

（6）应急响应

建立安全报警系统，通过安全中心收集第三方发现的安全问题，组织各部门对已经发现的安全问题进行修复、影响面评估、事后安全原因追查。

二、运维工作发展过程

阶段 1：早期的运维团队

早期的运维团队在人员较少的情况下，主要是进行数据中心建设、基础网络建设、服务器采购和服务器安装交付工作。几乎很少涉及线上服务的变更、监控、管理等工作。这个时候的运维团队更多的属于基础建设的角色，提供一个简单、可用的网络环境和系统环境即可。

阶段 2：随着业务产品的逐渐成熟

随着业务产品的逐渐成熟，对于服务质量方面就有了更高的要求。这个时候的运维团队还会承担一些服务器监控的工作，同时会负责 LVS、Nginx 等与业务逻辑无关的 4/7 层运维工作。这个时候服务变更更多的是逐台的手工操作，或者有一些简单批量脚本的出现。监控的焦点更多的在服务器状态和资源使用情况上，对服务应用状态的监控几乎很少，监控更多的使用各种开源系统如 Nagios、Cacti 等。

阶段 3：由于业务规模和复杂度的持续增加

由于业务规模和复杂度的持续增加，运维团队会逐渐划分为应用运维和系统运维两大块。应用运维开始接手线上业务，逐步开展服务监控梳理、数据备份以及服务变更的工作。随着对服务的深入，应用运维工程师有能力开始对服务进行一些简单的优化。同时，为了应对每天大量的服务变更，我们也开始编写各类运维工具，针对某些特定的服务能够很方便的批量变更。随着业务规模的增大，基础设施由于容量规划不足或抵御风险能力较弱导致的故障也越来越多，迫使运维人员开始将更多的精力投入到多数据中心容灾、预案管理的方向上。

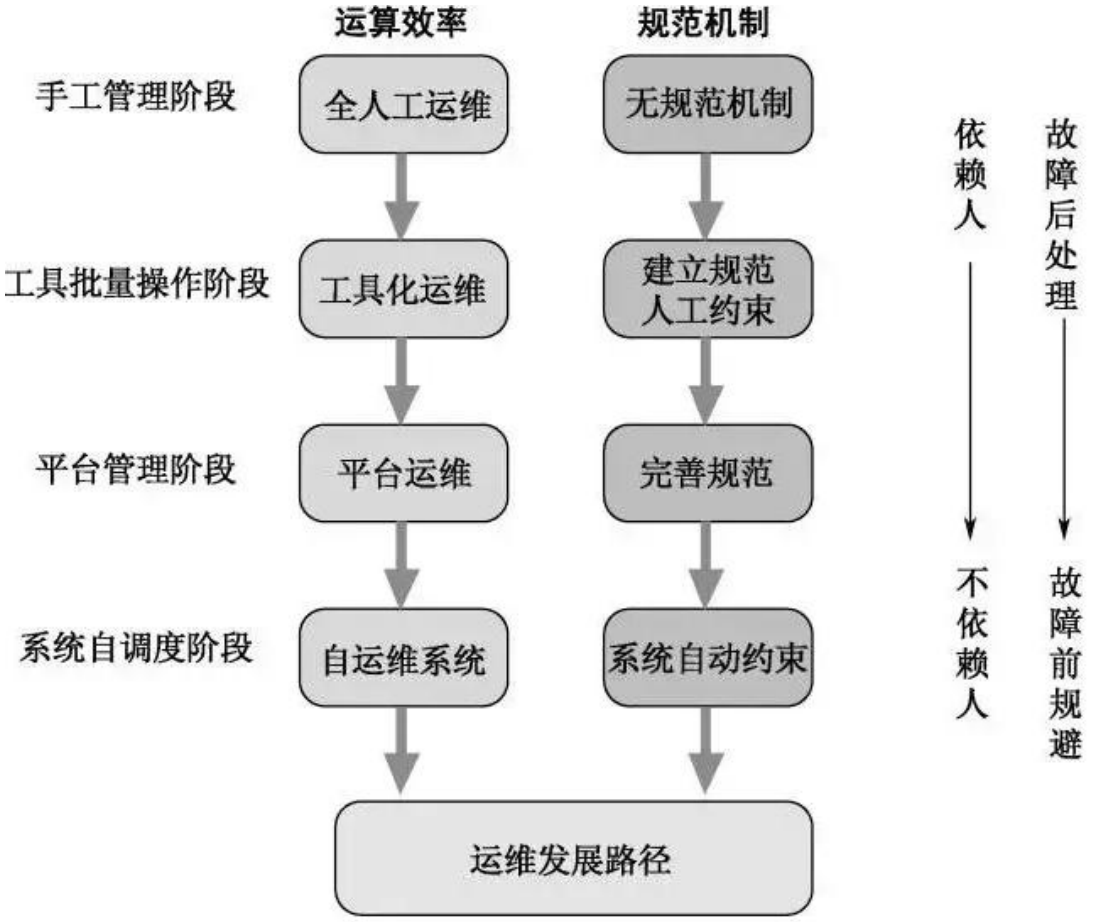
阶段 4：业务规模达到一定程度后

业务规模达到一定程度后，开源的监控系统在性能和功能方面，已经无法满足业务需求；大量的服务变更、复杂的服务关系，以前靠人工记录、工具变更的方式不管在效率还是准确性方面也都无法满足业务需求；在安全方面也出现了各种大大小小的事件，迫使我们投入更多的精力在安全防御上。逐渐的，运维团队形成之前提到的 5 个大的工作分类，每个分类都需要有专精的人才。这个时候系统运维更专注于基础设施的建设和运维，提供稳定、高效的网络环境，交付服务器等资源给应用运维工程师。应用运维更专注于服务运行状态和效率。数据库运维属于应用运维工作的细化，更专注于数据库领域的自动化、性能优化和安全防御。

运维研发和运维安全提供各类平台、工具，进一步提升运维工程师的工作效率，使业务服务运行得更加稳定、高效和安全。

4 个阶段的划分

我们将运维发展过程划分为 4 个阶段，如图 1-2 所示



手工管理阶段：

业务流量不大，服务器数量相对较少，系统复杂度不高。对于日常的业务管理操作，大家更多的是逐台登录服务器进行手工操作，属于各自为战，每个人都有自己的操作方式，缺少必要的操作标准、流程机制，比如业务目录环境都是各式各样的。

工具批量操作阶段：

随着服务器规模、系统复杂度的增加，全人工的操作方式已经不能满足业务的快速发展需要。因此，运维人员逐渐开始使用批量化的操作工具，针对不同操作类型出现了不同的脚本程序。但各团队都有自己的工具，每次操作需求发生变化时都需要调整工具。这主要是因为对于环境、操作的规范不够，导致可程序化处理能力较弱。
此时，虽然效率提升了一部分，但很快又遇到了瓶颈。操作的质量并没有太多的提升，甚至可能因为批量执行而导致更大规模的问题出现。我们开始建立大量的流程规范，比如复查机制，先上线一台服务器观察 10 分钟后再继续后面的操作，一次升级完成后至少要观察 20 分钟等。这些主要还是靠人来监督和执行，但在实际过程中执行往往不到位，反而降低了工作效率。

平台管理阶段：

在这个阶段，对于运维效率和误操作率有了更高的要求，我们决定开始建设运维平台，通过平台承载标准、流程，进而解放人力和提高质量。这个时候对服务的变更动作进行了抽象，形成了操作方法、服务目录环境、服务运行方式等统一的标准，如程序的启停接口必须包括启动、停止、重载等。通过平台来约束操作流程，如上面提到的上线一台服务器观察 10 分钟。在平台中强制设定暂停检查点，在第一台服务器操作完成后，需要运维人员填写相应的检查项，然后才可以继续执行后续的部署动作。

系统自调度阶段：

更大规模的服务数量、更复杂的服务关联关系、各个运维平台的林立，原有的将批量操作转化成平台操作的方式已经不再适合，需要对服务变更进行更高一层的抽象。将每一台服务器抽象成一个容器，由调度系统根据资源使用情况，将服务调度、部署到合适的服务器上，自动化完成与周边各个运维系统的联动，比如监控系统、日志系统、备份系统等。通过自调度系统，根据服务运行情况动态伸缩容量，能够自动化处理常见的服务故障。运维人员的工作也会前置到产品设计阶段，协助研发人员改造服务使其可以接入到自调度系统中。
在整个运维的发展过程中，希望所有的工作都自动化起来，减少人的重复工作，降低知识传递的成本，使我们的运维交付更高效、更安全，使产品运行更稳定。对于故障的处理，也希望由事后处理变成提前发现，由人工处理变成系统自动容灾。

三、运维的烦恼

如前一章所描述那样，随着业务和用户规模越来越大，公司对业务的稳定和质量开始重视起来，这时候公司才意识到需要专职的运维人员介入，而此时的业务系统已经变得非常庞大和复杂。此外，由于互联网产品快速试错的特点，服务架构也在不断地快速变化。

产品研发早期缺少相应的规范和标准，服务的部署方式、启停方式、配置和日志格式等都不统一，服务与服务之间的关联关系错综复杂，服务的各个环节都缺少监控；服务之间的耦合度很高，经常会由于一个小模块的崩溃，导致整个业务系统拒绝服务。

这个时候运维人员更像是保姆、消防员和拆弹专家。运维人员需要细心地呵护服务，让其健康地成长，就像照顾婴儿一样；成长中的服务，经常由于各种不规范带来的历史原因，出现很多意想不到的突发事情，这时候运维人员需要第一时间响应，进行业务的紧急恢复，类似消防员的角色；在日常的服务管理过程中，一个操作顺序或命令的错误，有可能直接让服务中断，这时候运维人员就像拆弹专家，既要细心大胆，又要有耐心，在危机时刻能够快速处理，做出正确决策。

运维人员需要处理各种突发的服务故障，在早期缺少统一规范、缺少业务监控、基础设施不成熟以及业务不断快速变化的时候，运维人员几乎每天都在忙于应付各种大大小小的服务故障。如图 2-1 所示是一个真实案例中，某个月统计到的服务故障分类和占比。

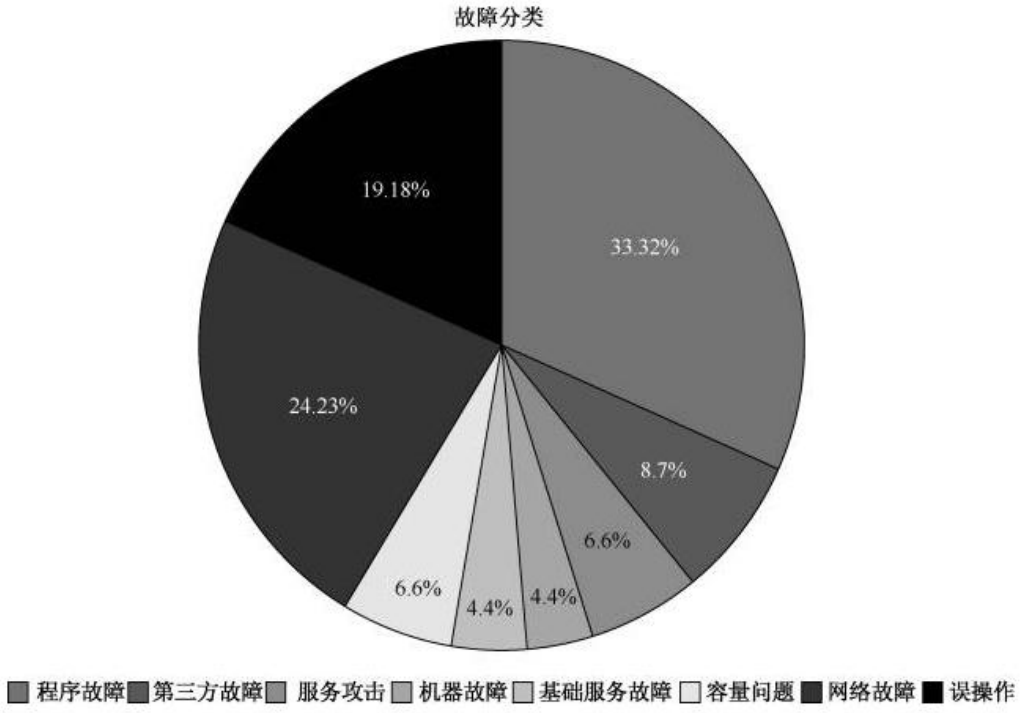


图2-1 服务故障分类和占比

根据监控项的重要程度对告警进行分级是一个很好的实践，Disaster 级别优先级最高，需要立即处理。Warning 级别需要 24 小时内完成处理，如图 2-2 所示。Warning 级别大多数是 CPU、内存、硬盘资源超限预警。详细的报警级别定义和划分，请参见后面监控章节的报警分级部分。

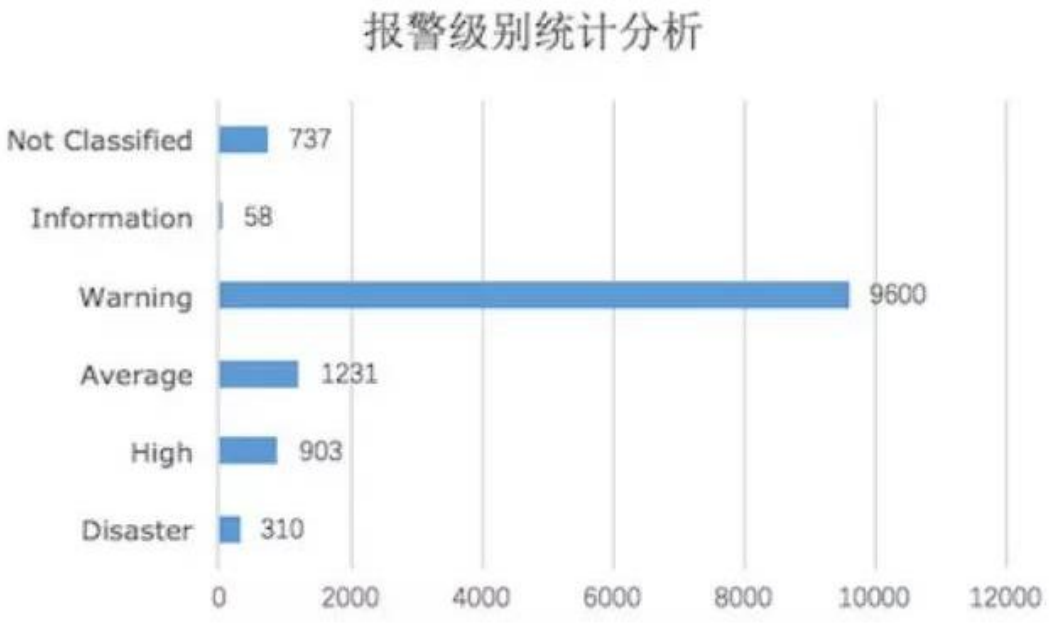


图2-2 各报警级别占比

注：根据监控项的重要程度对告警进行分级是一个很好的实践，P0 级别优先级最高，需要立即处理。P3 级别需要 24 小时内完成处理，P3 级别大多数是 CPU、内存、硬盘类资源超限预警。详细的报警级别定义和划分，请参见后面监控章节的报警分级部分。

业务快速变化、缺少统一规范、缺少文档和培训，运维人员基本上是摸黑接手服务。线上服务经常会埋着各种奇奇怪怪的坑，每一次服务变更都如履薄冰。

案例 1

服务上下游处理超时时间不匹配，上游服务的超时时间设置为 5 毫秒，而下游服务的超时时间却设置成了 10 毫秒，下游服务还在正常处理请求中，可上游服务却因达到超时设置而将本次请求丢弃了，最终客户端不断重试，导致服务器端压力增大。

一个真实案例中，遇到过上下游服务超时时间单位不一致的情况，因为系统中的各个服务是不同的研发人员负责的，在联调过程中忽略了一些问题，导致上游服务使用秒作为超时时间单位，下游服务却使用了毫秒。某次下游单机故障时，运维人员发现上游容错机制完全无效，依然导致其堆积了大量请求，最终影响服务整体性能。经过了较长时间的追查，才发现是由于超时时间单位问题引起的。

由于缺少规范化，给运维带来了无形的风险，而且故障定位也比较困难。

案例 2

集群服务是多台服务器共同完成一个任务，它们之间的调用关系是通过在程序配置文件中配置 IP 地址或服务器主机名来宣告的。由于 IP 地址的易读性较差，我们一般会使用内网 DNS 提供的主机域名。但有些研发人员却通过修改本机/etc/hosts 文件的形式自定义域名解析。

这样的修改，使得对目标域名的解析是维护在每台服务器上的，这将极大增加运维管理的风险。试想 100 台相互存在调用关系的服务器，每台服务器上都需要维护与其他多台服务器的域名关系。当出现服务变更、故障处理、服务迁移时，需要所有上下游服务配合变更，带来很高的操作风险和复杂度。

运维属于技术线的末端（见图 2-3），产品研发、测试、上线后将持续不断地在线上运行着。互联网产品很少有产品下线的情况，经常会出现某个产品的产品经理、研发工程师、测试工

程师都没有了，而这个产品依然还有运维人员在维护，持续提供服务。上游引入的任何缺陷，最终都由运维去承担，上游往往无法感受到运维的压力。随着业务的增长、服务与主机数量的增加，产品各个阶段的缺陷会被进一步放大，运维压力也越来越大。

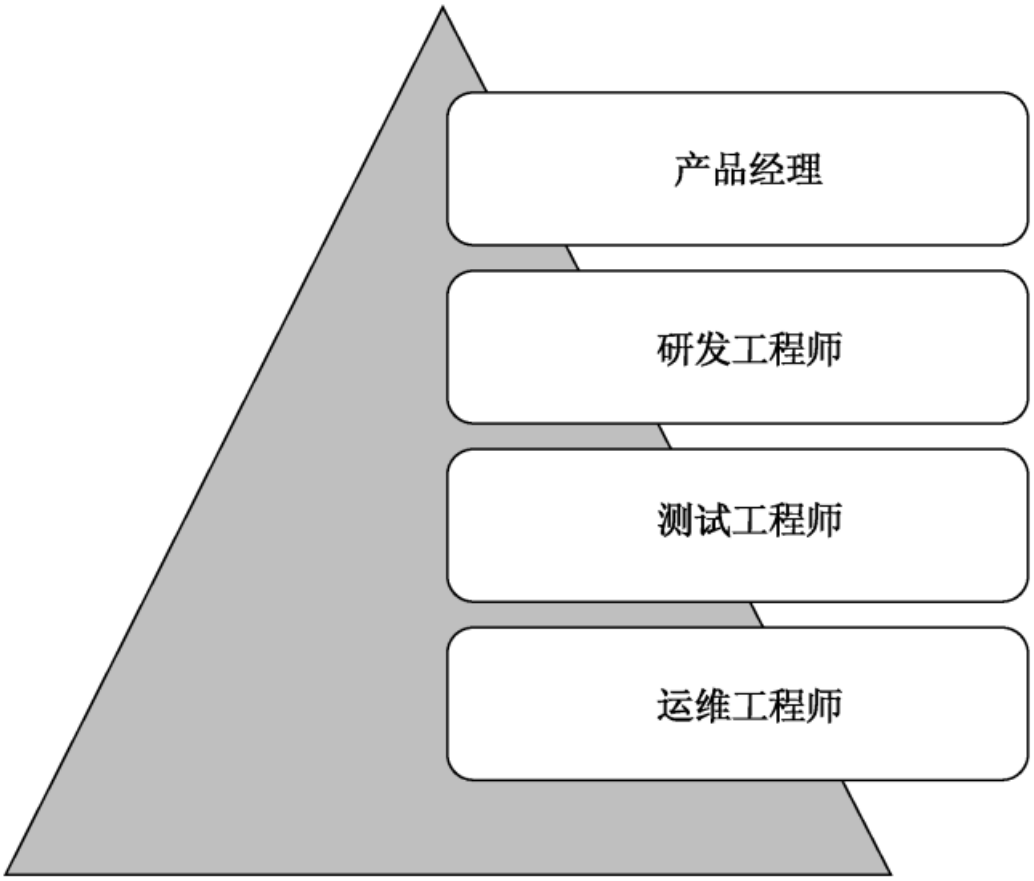


图2-3 运维属于技术线的末端

手工操作是初期运维团队的主要方式，渐渐的会形成一些工具或者系统，但都比较零散，适用场景较小，无法产生规模化。运维批量化和自动化所需要的信息非常少，这些信息基本上都靠人工录入，有哪些 IDC，放置了什么服务器，服务器部署了什么服务，这些信息都没有自动采集和联动，无法给自动化系统提供必需的基础信息。运维的重复性工作非常多，又较多属于手工操作，不仅效率低，而且手工操作带来的失误率也比较多，几乎无法消除。

运维承受来自于外部不断增长的业务压力，以及快速发展中引入的各种缺陷。同时又面对内部生产力低下，导致工作效率低下和误操作较多的现状。运维是一个比较尴尬的工作，属于技术线的末端，人力、技术和资源的投入也属于末端。运维不出故障是正常，任何由于资源不足、基础设施不稳定、人员误操作导致的问题，都会被业务部门投诉。不过近年来，运维工作的价值越来越被大家认可，运维支持能力成为公司的核心技术竞争力之一。

运维工作需要从两个方向去解决上述提到的问题：提高内部运维效率和降低外部运维压力。

经过统计，运维工作中占比最多的是服务变更、监控管理、容量管理和故障处理。我们需要开发运维工具和平台，在运维数据准确的前提下让所有的工作尽量自动化起来。制定相关的标准和流程，运维人员在项目设计阶段就参与进来，进行设计评审，让研发人员交付的项目符合运维准入的要求。同时，让研发人员使用运维相关的工具，使研发、测试、上线阶段的部署行为一致，监控策略一致，且被测试验证过。

运维标准不是凭空制定出来的，需要满足运维自动化相关工具的最低要求。符合运维标准的产品，能够更加方便地进行一键部署，与监控联动等，这样才使研发人员有动力往运维标准靠拢，更积极地使用运维工具，我们的标准和工具才能进一步得到。

四、运维的未来

DevOps 的出现

相对于瀑布开发模式，敏捷开发过程的一个基本原则就是以更快的频率交付最小化可用的软件。高频率的部署，每天数百个版本的发布，经常会由于运维部署自动化程度的不足，导致部署任务堆积在运维人员的面前。

对于研发人员而言，线上服务运行环境对其属于黑盒，研发、测试和上线时的行为不一致，研发和运维之间的沟通错位，会造成各种部署问题。

为了解决传统意义上的研发行为和运维行为存在的脱节现象，提高持续交付的效率，DevOps 的理念应运而生。为了适用与 DevOps 相关的快速部署节奏，ITIL 流程的很多方面，特别是围绕着变更、配置和发布流程方面，需要将所有过程尽量自动化起来。

伴随着 DevOps 的理念，涌现出一批优秀的开源软件，比如 Jekins、Puppet、Chef、SaltSatck、Docker 等。Jekins 属于持续集成的自动化构建工具；Puppet、Chef、SaltStack 属于配置管理和任务执行类工具，方便我们快速同步变更到所需要的环境中。

Docker 是近两年来最火的开源项目，它在 Cgroup、LXC 基础之上封装，基于 Linux 内核支持的 NameSpace 进行资源隔离，实现了进程级虚拟化。

早期我们考虑服务整体部署的时候，抽象认为每台服务器类似一个终端设备，每个需要被部署的服务是其上安装运行的 App 应用。为了达到服务整体部署，不破坏系统纯净的环境，我们制定了一系列部署要求。比如，要求服务所依赖的相关组件自包含、自解决；制定线上目录规范，进行环境隔离；要求所有服务必须有统一的启停接口，方便运维人员或系统进行控制管理等。

在 Docker 出现后，这些部署需要考虑的前置条件都迎刃而解，Docker 逐渐成为我们整个运维体系中一个不可或缺的关键组件，也是部署自动化、动态调度部署的前提。

研发人员通过 Docker 进行服务封装，能够很方便地在开发、测试和线上环境中部署运行服务，达到部署行为的幂等。在此过程中，运维人员不需要过多的参与，只需要把 Docker 容器放置在合适的地方启动即可；在有 IaaS、PaaS 或者自有调度系统支持的情况下，甚至部署 Docker 容器、启动容器和监控服务的工作都可以由系统自动完成，整个部署工作不需要运维人员参与。

云服务的普及

近些年来，随着云服务概念的普及、云服务厂商的持续投入、技术的不断发展，云服务的单位成本在持续下降，云服务已经成为一种不可阻挡的趋势。这给传统的 IT 建设理念，造成了一定的冲击。

传统的应用部署，需要兼顾应用程序、数据、运行时环境、中间层、操作系统、虚拟化、服务器、存储、网络等很多方面，单位建设成本和维护成本很高，同时对专业人才的需求更多、要求更高，无形中增加了业务运行的成本支出。图 5-1 描述了传统 IT 建设和 IaaS、PaaS 所关注的不同层次。

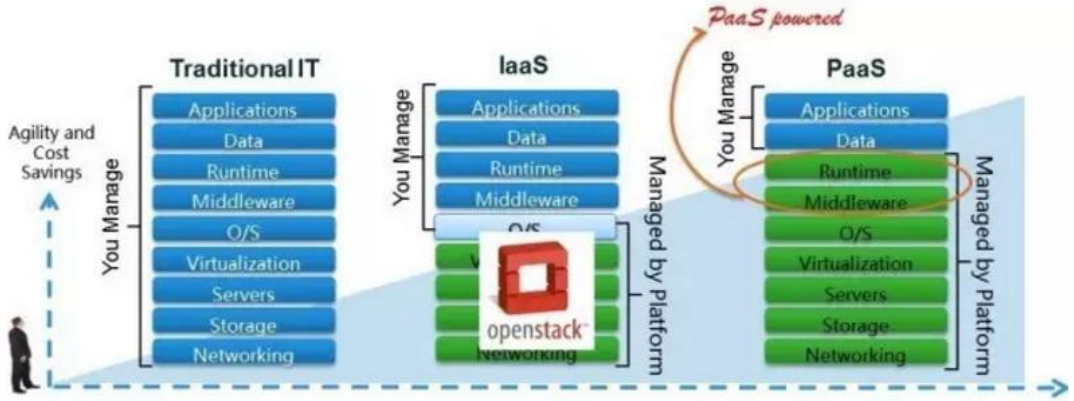


图5-1 传统IT和IaaS、PaaS方面的对比

IAAS

（基础设施即服务），在传统应用部署的基础上，将 OS 以下的层次都负责起来，即操作系统、资源虚拟化、服务器、存储、网络，减少运维工作量；OpenStack 作为领先的开源解决方案，也是我们运维体系建设的重要方向之一。提供商业服务的 AWS，也是属于 IaaS 平台典型的代表。

安全问题是影响我们使用云服务的关键因素，和传统运维不一样，网络设备、宿主机等资源的权限都在云服务厂商手里，网络流量的出入也需要经过他们的设备，云服务厂商自身的安全规范和审计还比较薄弱。

随着 HTTPS 的大量使用、VPC 的成熟、云服务厂商自身安全加固等，这些安全因素也在逐渐减小。当然，由于所有用户使用虚拟机共享宿主机，虚拟机被攻陷获取宿主机的安全问题还依然存在，比如 2015 年 5 月份披露的毒液漏洞（VENOM，CVE 编号 CVE-2015-3456），攻击者可以在有问题的虚拟机中进行逃逸，获取宿主机代码执行的权限。

现在也有一些云服务厂商，比如金山云也提供了混合云的服务，我们可以把非关键服务部署在云服务上，把关键服务部署在物理环境中，由我们控制物理设备的权限，通过公网或者私有专线进行通信，缓解云服务暂时还不能满足的安全需求。

云服务的出现，我们不需要再关注 OS 层面以下的问题。随着云服务的逐渐成熟，能够提供足够的资源储备和交付效率，我们不需要投入大量的人力在机房建设、服务器采购和网络管理方面，也不需要储备更多的资源应对突发业务、网络攻击等。对于公司而言，资源得到进一步优化，不需要那么多的基础运维人员，而且效率比以往更高了。

PAAS

（平台即服务），在传统应用部署的基础上，将应用的运行时环境、中间层通过规范化的方式给管理起来，并实现容量的自动伸缩。本质上，PaaS 是作为一种应用部署的规范，并通过相应的机制来保障该规范的实施，以及进一步地实现资源的动态调度，达到容量自动伸缩的目的。

CloudFoundry 属于开源 PaaS 平台的典型代表，只要研发程序遵从一定的约束条件，服务的运行发布、扩容和故障容灾都由 PaaS 平台统一负责。PaaS 由于具有强约束性，主要适用于简单业务，但不能满足比较复杂的业务架构。

我们可以根据 PaaS 的特性，通过 Docker 容器将业务进行封装，使业务达到可随意部署和资源隔离的程度，并参考 Borg 的设计思路定制动态调度系统，通过系统调度服务的部署和监控，减少对应用运维人员的依赖。

SAAS

云服务厂商还会提供其他各种类 SaaS（软件即服务）的服务，比如 CloudWatch、EMR（Elastic MapReduce）、RDS（Relational Database Service）、CloudFront 等，进一步降低对运维人员的依赖，运维人员不需要搭建和运维相关的基础设施或服务。

比如 RDS 提供丰富的数据库主从搭建、数据迁移、慢查询监控等功能，不需要数据库运维人员投入过多精力在数据库日常运维方面，可以把更过的精力投入在数据库设计和优化方面。StatHat 属于 SaaS 类的云监控服务，我们自研的监控系统是参考它设计的。

网络虚拟化

在传统网络架构中心，根据业务需求部署后，如果发生任何变更，都需要重新修改网络设备（如交换机、防火墙）的配置。这是一个非常痛苦的过程，并且不同品牌网络设备的 OS 都各不相同，统一配置的难度可想而知。

在移动互联网瞬息万变的今天，高稳定和高性能的网络已经不足以支持业务的需求，灵活性和敏捷性更为关键。传统网络模式的问题在于它的封闭性和自治性很难与应用直接产生关系。

随着 SDN（Software Defined Network）技术的成熟，可以让其变得更轻而易举。SDN 提出将控制层面和数据层面分离，通过集中或分布式控制器统一管控，通过 OpenFlow 协议提供网络的可编程能力，让用户可以在网络上定义各种应用程序，通过软件来定义逻辑上的网络拓扑，以满足对网络资源的不同需求，而无须关心底层网络的物理拓扑结构。

在未来，也许网络设备也类似一台台服务器，由软件控制和管理，不需要过多依赖网络运维人员。

总结

运维的未来是充满危机和挑战的，从当前虚拟化和云计算的发展趋势来看，未来的互联网服务一定是依托在云端。通过云服务厂商提供的各类服务组件，结合 DevOps 的理念，将运维所涉及的工作串联起来、自动化起来。

运维人员的工作逐渐由体力密集型向脑力密集型转变，不再需要大量的运维人员从事那些基础建设、服务变更、故障处理等烦琐的体力工作。

运维工作更大的方向是提供平台化的运维产品，提供运维数据的可视化、运维工作的自动化，由操作、响应类转变为优化、规划类的工作内容。甚至连传统的硬件设备管理，也逐渐地通过软件层面去实现，更加的高效和自动化。

作为运维人员，已经不能和以前一样，只关注某个产品、某个领域的专业技能运维，我们需要关注技术的变化趋势，拥抱变化，做好运维转型，成为云服务建设或使用的专家，成为业务规划或性能优化的专家，或许……

ZYXW、参考

《运维之下》
第一章、互联网运维工作
http://mp.weixin.qq.com/s?__biz=MzA3MzYwNjQ3NA==&mid=2651297197&idx=1&sn=44357825f3474120c911f14f088e3f48&chksm=84ff4188b388c89ee1f6edab38ea3337ef21dd83cda73e9b0a3f1673dea856560ab76a6ed77f&mpshare=1&scene=1&srcid=1010awzKRWv62vLr9wxQkTTC#rd
第二章、运维的烦恼
http://mp.weixin.qq.com/s?__biz=MzIxMzIyNDkyMw==&mid=2654107969&idx=1&sn=66a8feefe35f7bb67a5f36919a9fbef9&scene=21#wechat_redirect
第五章、运维的未来
http://mp.weixin.qq.com/s?__biz=MzIxMzIyNDkyMw==&mid=2654108056&idx=1&sn=7de0d33032cf15381663b8ee6b168db9&scene=21#wechat_redirect