分布式系统概要

Aneesh Khole
计算机科学系
（MMCOE）
Marathwada Mitra Mandal 工程学院
工程
（SPPU）
浦那，印度
aneeshkhole2019.comp@mmcoe.edu.in

Atharva Thakar
计算机科学系
（MMCOE）
Marathwada Mitra Mandal 工程学院
工程
（SPPU）
浦那，印度
atharvathakar2019.comp@mmcoe.edu.in

Avadhoot Kulkarni
信息技术系
（隶属 PVGCOET）
浦那维迪亚尔蒂格里哈工程与技术学院
工程与技术
（SPPU）
浦那，印度
avadhootkulkarni06@gmail.com

Hrithik Jadhav
机械系
（MMCOE）
Marathwada Mitra Mandal 工程学院
工程
（隶属 SPPU）
浦那，印度
hrithikjadhav2019.mech@mmcoe.edu.in

Shreyas Shende
计算机科学系
（MMCOE）
Marathwada Mitra Mandal 工程学院的
工程
（隶属 SPPU）
印度浦那
shreyasshende2019.comp@mmcoe.edu.in

Varad Karajkhede
计算机科学系
（MMCOE）
Marathwada Mitra Mandal 工程学院
工程
（SPPU）
印度浦那
varadkarajkhede2019.comp@mmcoe.edu.in

摘要— 计算机系统多年来不断演进，从大型、单用户、运行缓慢且价格昂贵的机器，发展到多用户、运行快速、价格低廉且体积小巧的机器。多用户计算机网络的应用催生了一种称为分布式系统的新型计算范式。分布式系统被视为由相互依赖的网络通信节点和计算节点组成的软件。这种范式通过将各种计算机相关任务去中心化并分配给多个互连的计算机节点，在实现高性能的同时，也保持了高效率。尽管分布式系统多年来已被证明大有裨益，但仍存在一些设计缺陷、安全问题和挑战。本文旨在阐述这些问题、挑战和安全问题，同时探讨多年来为解决这些问题而开发的各种方案。本文还简要介绍分布式系统的组件及其工作原理。

关键词— 异构性、并发、可扩展性、HTCondor、Entropia、加密。

一、引言

1.1 什么是分布式系统?

多年来，人们提出了多种分布式系统的定义，但这些定义既不令人满意，彼此之间也不一致。在 Coulouris 提出的若干定义中，有一种将分布式系统定义为“一种系统，其中位于联网计算机上的硬件或软件组件通过消息传递”来进行通信并协调各自的操作。[1]

另一个可用于简要概述分布式系统的定义是：

分布式系统是一种计算环境，其中众多组件分布在网络中的多个计算设备上。这些设备分担工作，并通过通信与协调彼此的工作，使最终用户感觉它们是一个统一的整体系统。分布式系统可以由不同的配置组成，例如大型机、计算机、工作站和小型计算机。

这一定义突出了分布式系统的两个主要特征。首先，分布式系统由各种计算机组件构成，其中既可以包括软件组件，也可以包括硬件组件。这些组件彼此独立，计算元素称为节点。第二个特征是，用户感觉自己仿佛独自在使用该系统。

1.2 为什么需要分布式系统?

现代计算之所以成为可能，离不开分布式系统。分布式系统对于云计算服务、无线网络和互联网的性能至关重要。分布式系统能够提供更高的可用性。没有分布式系统，这些进步便不可能实现。在大多数情况下，分布式系统甚至应用于复杂程度远不及整个电信网络的企业岗位。分布式系统克服了单体系统的缺点，带来了更高的可扩展性和更出色的性能。

分布式系统能够利用其他计算设备和进程，提供单个系统难以升级实现的功能。定期备份服务器和应用程序（包括位于远程位置的服务器和应用程序）是一项必不可少的任务。需要恢复时，可以通过请求相关数据段，从异地备份节点获取所需数据。

计算设备上的一切实际上都依赖于分布式系统的运行。

II. 分布式系统的架构与运行

2.1 分布式系统的组成要素

分布式系统的三个组件包括主系统控制器、系统数据存储和数据库。在非集群环境中，可以使用用户界面和次级控制器等可选组件。

2.1.1 主系统控制器

主系统控制器是分布式系统中唯一存在的控制器，负责管理所有事务，并控制服务器请求在整个系统中的分派。主系统控制器会跟踪整个系统的状态信息，并控制所有其他系统组件。 主系统控制器会自动安装执行服务和邮箱服务。

2.1.2 次级控制器

次级控制器充当进程或通信控制器。它负责调节服务器处理请求的流量，并管理系统的转换负载。它还负责管理系统与增值网络或贸易伙伴之间的通信。

2.1.3 用户界面客户端

用户界面客户端也是系统中的一个附加元素，其任务是向用户提供重要的系统信息。用户界面控制器不属于集群环境，也不在与控制器相同的计算机上运行。它的功能包括监控和控制系统。

2.1.4 系统数据存储

无论是否采用集群，系统的数据存储通常都位于磁盘存储库中，并且每个系统只有一个数据存储，用于存放所有共享数据。在非集群系统中，数据存储可以位于单台计算机上，也可以分布在多台计算机上，但所有计算机都应能访问该数据存储。

2.1.5 数据库

对于分布式系统，所有数据都存储在关系型数据库中。数据存储从数据库中找到数据后，会将数据共享给多个用户。所有数据系统都使用关系型数据库，因此多个用户可以同时使用相同的信息。

2.2 分布式系统的架构

该`系统级架构` 关注整个系统以及分布式系统各组件在多台机器上的部署位置。分布式系统分为四种不同的架构模型

1. 客户端-服务器——客户端-服务器架构由客户端和服务器组成。客户端从服务器检索数据，然后对其进行格式化并显示给用户。最终用户也可以在客户端进行修改，并将修改发送回服务器以使其永久生效。 客户端-服务器架构有一个标准设计特征：集中式安全。

2. 三层架构——三层架构中的客户端是无状态的。为简化应用程序部署，客户端信息不保存在客户端，而是保存在中间层。三层架构最常用于 Web 应用程序。

3. N 层架构——在 N 层架构中，中间层会与另一项服务交互以获取信息。当应用程序或服务器需要将请求转发至网络上的其他企业服务时，通常会使用 N 层架构。

4. 对等网络——对等网络（也称为 P2P 网络）基于去中心化系统的概念运行。 系统中没有额外的机器用于提供服务或管理资源。系统中的机器称为对等节点，它们的职责均匀分布。客户端和服务器都可以使用这些机器。

2.3 分布式系统的工作原理

分布式系统随着当今分布式系统的用例不断发展，这些系统旨在通过互联网和云运行。我们可以通过以下示例来说明分布式系统的工作方式。整个过程从一项任务开始，例如渲染视频。随后，视频编辑器会管理这项任务，并将工作拆分成多个部分。系统会使用多台计算机（称为节点），每个节点负责处理视频中的一帧。这些节点独立处理单帧；完成该帧的渲染后，视频编辑器会分配一帧新的画面供其处理。这个过程会持续进行，直到所有帧都渲染完毕。此外，得益于分布式系统，节点数量不受限制。节点数量与完成渲染所需的时间直接相关。因此，使用更多节点意味着能更快完成任务。下图对此进行了说明。

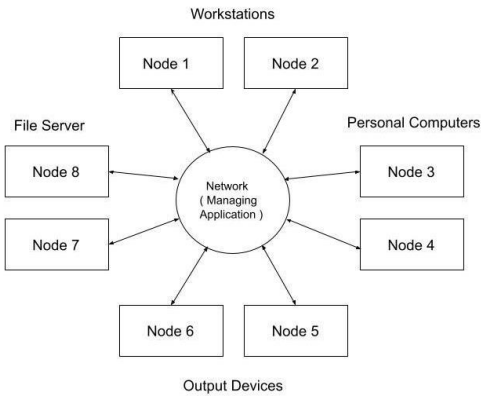

图 1. 分布式系统示意图

所有节点都通过网络相连，并在进程运行期间相互通信。这些节点协同工作以完成进程，同时也能够执行各种操作。如今，人们使用多种模型和架构，例如客户端-服务器和点对点系统。这两种架构产生的输出相同，但实现输出的方式不同。上面的示例说明了分布式系统的工作方式。

III. 安全性

分布式系统中的安全性大致可分为两部分。第一部分用于指导用户或进程之间的通信，这些用户或进程可能驻留在专用计算机上。确保通信安全的重要机制是安全信道。第二部分涉及授权，其目的是确保进程仅获得分布式系统中相应资源的访问权限。它关注安全信道和访问控制，这需要借助相关机制来分发密码密钥，同时还需要提供将用户添加到系统或从系统中移除的机制。这些主题通过所谓的安全管理来加以讨论。

计算机中的保护与可信性的概念密切相关。通俗地说，可信的计算机系统是我们相信能够提供服务的系统。计算机系统的可信性涵盖多个方面，例如可访问性、稳定性、安全性和可维护性。然而，要确保系统完全值得信赖，还必须考虑保护敏感信息和维护数据完整性等因素。安全计算机系统中发生的不当更改必须能够被检测并恢复。任何计算机系统最重要的资产是其硬件、软件程序和统计数据。

3.1 安全威胁

1. 截获

截获这一概念指出，未授权实体已获得对某项服务或某些信息的访问权限。

2. 中断

中断是指服务或数据变得难以获取、无法使用、遭到破坏等的状态。根据`拒绝服务攻击`，第三方非法企图使其他实体无法访问某项服务，这是一种被认定为中断的安全威胁。

3. 篡改

篡改是指未经授权地破坏数据，或篡改服务，使其不再符合正式规范。

4. 伪造

伪造是指创建通常不存在的额外数据或活动的状态。

3.2 安全需求

1. 机密性

这意味着未经授权的个人无法访问数据。只有经过身份验证过程后，相关主管部门才能访问数据。

通过加密数据来维护机密性。

2. 完整性

它可以避免数据遭到未经授权的更改，并检测是否发生了任何更改。许多身份验证算法都用于此类验证流程。这有助于确保只有授权人员才能修改任何数据。

3. 可用性

它确保只有相关授权人员才能访问数据。这意味着其他人无法访问数据并对其进行修改或更改，因此这是一项安全性要求。

3.3 安全机制

1. 加密

加密是计算机安全的基础。加密会将数据转换为攻击者难以理解的实体。加密提供了一种实现数据机密性的方法，并使我们能够检查数据是否已被修改。基本方法如下：

i. 传统加密：发送者使用同一个密钥加密消息，接收者使用该密钥解密消息。这也称为对称加密。ii. 公钥加密：这里使用两个密钥来

进行加密，其中一个密钥公开，供任何人使用。这也称为非对称加密。

2. 身份验证

身份验证用于验证用户、客户端、服务器、主机或其他实体所声称的身份。对于

客户端，最基本的一点是，在服务以客户端为由开始执行任何工作之前，服务必须了解客户端身份。

3. 授权

客户端通过认证后，有必要检查该客户端是否有权执行所请求的操作。这意味着要检查客户端是否满足相关要求。

4. 审计

审计工具用于跟踪哪些客户端访问了哪些内容，以及以何种方式访问。尽管这种方法无法提供任何抵御威胁的保护，但审计日志对于调查安全违规事件以及据此采取措施应对攻击者而言，可能至关重要。

3.4 密码学在安全措施中的应用

密码技术的使用是分布式系统安全的一项重要基础。它无法确保系统的安全性，但却是构建安全分布式系统的重要组成部分。

这些技术的应用非常简单。
设想这样一种情况：发送者想要将一条消息传输给另一位接收者。发送者将同一条消息转换为无法理解的消息，并将其发送给接收方，以保护消息免受各种安全问题的影响。要以原始形式获取该消息，接收者必须对其进行解密。这类加密—解密方法通过由密钥参数化的密码学方法得以实现。在这里，原始消息称为明文，而其加密形式称为密文。

在传输密文消息时，我们需要防范三种不同的攻击。第一种可能的威胁是入侵者在发送者或接收者毫不知情的情况下截获消息。如果明文经过加密后无法使用适当的密钥进行解密，那么截获便毫无意义。入侵者只能看到无法理解的数据。第二种可能的攻击是修改消息或对其进行任何更改。虽然修改明文并不困难，但要修改已经加密的密文则更加困难，因为必须先对消息进行解密，才能有意义地修改它。此外，修改消息后进行的加密必须准确无误，否则接收方可能会发现消息已被某些未经授权的人员篡改。

最后一种可能的攻击方式是，入侵者自行插入加密消息，而接收者误以为这些消息是由发送者发送的。[2]

3.5 设计问题

分布式系统或任何计算机系统，都必须提供安全系统，以便落实种类繁多的安全准则。在实施通用用途的安全保护时，必须考虑几个关键的设计因素。在考虑应用程序（可能是分布式的）的保护时，可以采用三种不同的方法。

1. 第一种方法是着重保护与应用程序直接相关的数据和信息。这里的“直接”是指，无论对某个数据项目可能执行哪些功能，首要考虑的都是确保数据完整性。通常，这种安全机制会出现在数据库系统中，其中可以制定大量完整性约束，并在每次数据项目发生更改时进行例行检查。

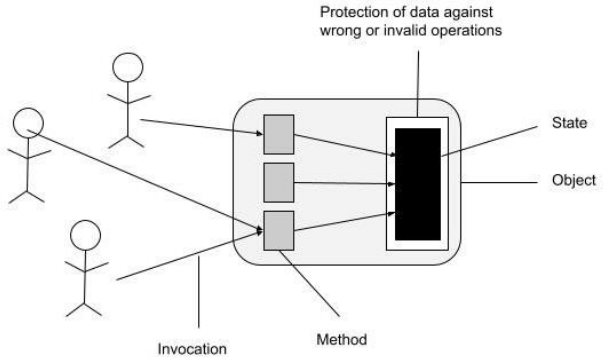

图2：防止数据遭受无效操作 *[2]*

2. 第二种方法是通过明确规定哪些人可以在何时调用哪些操作、访问哪些数据或资源，来实现安全性。在这种情况下，控制能力与访问控制方法密切相关。访问控制方法可应用于实体所提供的整个接口，也可应用于整个对象本身。因此，这种方法支持不同粒度的访问控制。

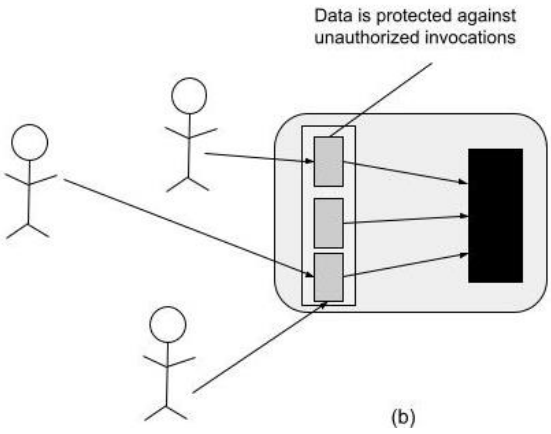

图3. 防止数据遭到未经授权的调用*[2]*

3. 第三种方法是直接针对用户采取措施，确保只有选定的人员能够访问应用程序，而不论他们想要执行哪些功能。实际上，控制的重点在于定义用户拥有的用例；一旦验证了用户的角色，就会授予或拒绝其访问某项资源。作为设计安全系统的一部分，因此有必要

描述人员可能担任的角色，并提供支持基于角色的访问控制的机制。

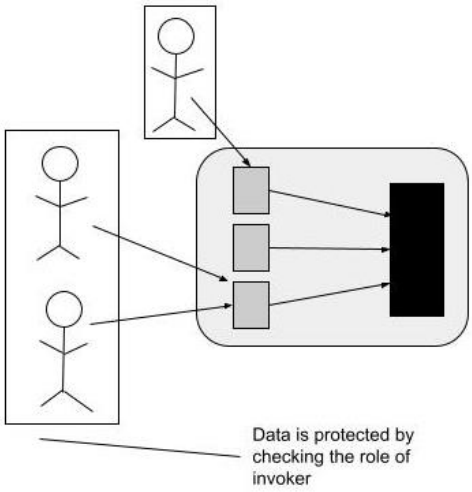

图 4. 通过检查调用者的角色保护数据*[2]*

第四. 分布式系统面临的挑战

设计分布式系统并非易事。要实现理想的系统，需要克服诸多挑战。下图列出了分布式系统面临的主要挑战。

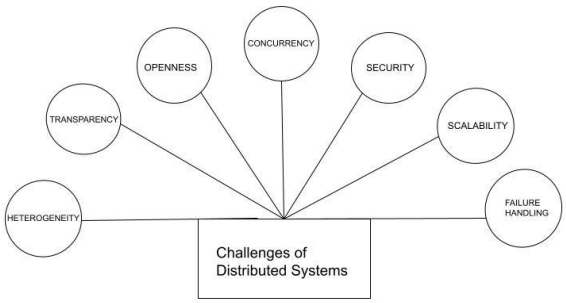

图 5. 分布式系统面临的挑战

4.1 异构性

为多种服务提供访问途径，并在各种各样的计算机上执行应用程序。通过互联网，客户端和最终用户可以使用、运行和访问涵盖种类繁多的计算机网络的工具和实践。
硬件设备：计算机、平板电脑、手机、嵌入式设备等。

不同的操作系统版本，例如 Windows 操作系统、Linux 操作系统、Mac 和 Unix。可使用的不同网络类型包括局域网、互联网、无线网络和卫星连接。可以使用 Java、C/C++、Python 和 PHP 等编程语言。要实现通信，必须采用并接受标准化的互联网协议。中间件是指一种软件层，它提供编程抽象，同时还模拟核心网络、编程语言和操作系统硬件的异构性。可移植代码就是能够从一台计算机系统传输到另一台计算机系统，并从一台计算机重新分发的程序代码。Java Web Start 就是可移植代码的一个例子，而 Java 小程序现已弃用。

4.2 透明性

将分布式系统视为集中式系统的单一整体，而非独立元素的集合，这一理念是分布式环境中透明性的核心。分布式系统开发者力求尽可能隐藏系统的复杂性。透明性标准可能包括自定义权限、移除、并发、故障、持续时间和客户资源。透明性也被定义为对分布式系统中各组件的隔离，使系统被视为一个整体，而不是自主组件的集合。换句话说，分布式系统开发者必须尽可能隐藏系统的复杂性。完全透明性的目的，是彻底隐藏计算由分布式系统提供这一事实。

4.3 开放性

分布式系统中开放性的目标，是以便于配置或修改的方式设计网络。开发者通常需要在任意分布式节点中添加新功能，或修订、替换现有的任何部分；如果有基于某些标准协议的适当互操作性支持，这些操作就能轻松实现。已发布的内容被撤回或突然删除时，开放性问题会更加严重。此外，分布式开放系统通常没有中央权威机构，因为不同系统可能各有其仲裁方。例如，Meta、Tinder 等组织允许开发者通过其应用程序接口以交互方式构建软件。

4.4 并发

并发用于管理对资源的同步访问。它可以防止多个用户同时修改同一条记录，还会按照特定顺序组织事务，以便进行备份和恢复。例如，在拍卖过程中，多个参与者会对同一对象出价；同样，在分布式环境中，由于服务器可能会尝试访问共享资源，因此多个客户端可能会同时尝试访问共享服务。为了确保对象安全地处于同一位置，必须在临界环境中执行其功能，并协调进程，使其数据保持一致。这可以通过多种方式实现，例如使用信号量；信号量在许多应用程序中都得到了广泛应用。多个用户尝试访问同一组数据或共享资源时，会导致并发问题。如果没有同步，进程的最终输出就会受到显著影响或发生变化

进程会按照输入的顺序执行。

4.5 安全性

分布式系统中提供并维护的大多数信息资源对用户而言都具有很高的内在价值，因此其安全性至关重要。在使用公共网络时，安全性是分布式环境中最受关注的问题。信息资源的安全性包含三个方面：

1. 机密性——（防止信息泄露给未经授权的人员）；完整性——（防止信息被篡改或欺骗）。

2. 可用性——（防止对客户端资源访问途径的干扰）。分布式系统中必须确保可用性。加密是避免安全问题的方法之一。应确保只有经过授权且合法的用户才能访问资源、修改资源并执行操作。目前，世界上许多机构和组织都已设计并开发了采用分布式环境的系统，这些系统具备上述所有安全特性。

4.6 可扩展性

当系统缺乏处理资源或用户数量突然增加的能力时，就会遇到可扩展性问题。在这种情况下，必须有效利用架构和算法。系统需要根据规模、地理范围或管理需求进行扩展。

- 规模：规模是指需要处理的用户和资源数量。规模可能导致的问题包括过载。
- 地理范围：地理范围是连接用户与资源的距离。由于地理限制而产生的问题之一是通信可靠性。
- 管理：随着规模持续扩大，需要对分布式系统的节点进行控制。与管理相关的可扩展性问题会导致管理混乱及其相关困难。

4.7 故障处理

计算机系统有时可能会发生故障。当硬件或软件发生此类故障时，程序可能会产生错误结果，或在完成预期任务之前终止。处理此类故障极其困难。 分布式系统由一组自主计算设备组成，它们面向客户端呈现统一的形象，并作为一个协调一致、紧密集成的整体运行。 系统由各种组成部分构成，这些部分协同工作以实现特定目标。随着系统的纵向扩展和规模扩大，随着系统变得更加分布式，系统发生故障的可能性也会增加，从而导致可信性下降。换句话说，在分布式系统中，某些元素可能无法运行，而其他元素仍然正常工作。这种现象称为部分故障。由于时间具有非确定性，消息及其到达目的地所需的时间也具有非确定性，因此我们说部分故障是不可预测的。我们无法收到确认来判断系统究竟是成功还是失败。分布式系统中的部分故障包括节点崩溃或通信连接问题。

V. 分布式系统面临的挑战及解决方案

5.1 异构性解决方案

为了解决与异构性相关的问题，我们展示了分布式计算领域中选定的方法如何应对不同类型的异构性。尽管有许多用于共享计算资源的系统，本概述仍介绍了能够代表整个领域的若干框架。我们介绍了一个批处理工作负载管理系统（HTCondor）、一个基于项目的志愿者与网格计算系统（BOINC）、一种面向服务的桌面网格方案（Aneka）、一个企业网格（Entropia），以及一种基于库的分布式计算编程模型（libWater）。

本提案主要关注分布式系统的开发。以下任务相关事项不在考虑范围内，即可访问性及其性质。

我们概述了采用不同方法的各类框架。每个系统的侧重点各不相同，例如高吞吐量、安全性或应用集成。

下表展示了它们如何处理异构性的不同维度。

	Hardware	运营ting系统m	程序ming语言e	可访问性能力	Nature of Tasks
HTCondor	✗	✗	✗	✓	✓
BOINC	✓	✗	✗	✓	✗

Aneka	✗	✓	✗	✓	✓
熵 ia	✗	✗	✗	✓	✓
lib 水	✓	✓	✗	✗	✗

表1. 异构性维度 *[3]*

5.2 透明性解决方案

5.2.1 访问透明性

本地和远程访问方法之间不应有明显差异。换句话说，开放通信可以保持隐蔽。例如，从用户的角度来看，将远程控制服务作为打印机使用，应与使用本地打印机无异。从程序员的角度来看，访问远程对象可能与访问同一类别的本地对象无异。这种透明性包含两个方面：

在分布式访问与非分布式访问之间保持语法或机制一致性，并保持语义相同。由于远程语义非常复杂，尤其是故障处理方式，因此本地访问必须采用最小集合。远程访问并不总是表现得与本地访问相同，因为某些服务可能没有必要支持（例如，对于单因素分布式系统而言，从网络流量角度看，进行完整的全局搜索可能并无意义）。

5.2.2 位置透明性

系统拓扑结构的细节不应成为用户的困扰。系统中某个项目的位置可以不向用户或编辑器显示。这一点与开放访问不同，因为设计方法和访问方法都可以相同。文字可能无法提供任何空间。
并发与应用的用户应能够访问共享数据或项目，且不会受到其他用户的中断干扰。这要求采用更精密的分布式系统方案，因为与中央系统模拟相比，现实中的兼容性要求更高。例如，分布式打印服务应提供与中央系统相同的文件访问权限，避免打印过程中出现不可预测的干扰。出于可用性或性能方面的考虑而复制系统，不应影响用户，其中也包括应用程序编辑器。

5.2.3 复制透明性

如果系统为了可用性或性能而进行复制，就不应影响用户。为确保充分的开放性，我们也将应用程序编辑器视为用户。

5.2.4 故障透明性

如果软件或计算机组件发生故障，用户不应察觉。在分布式系统中，这一点可能难以实现，因为通信系统的底层部分可能发生故障，而且故障可能不会被报告。应尽可能通过与公共访问相关的程序提供明确的错误信息。然而，在分布式系统环境中发现错误时，可能无法维持可访问性。使系统能够隐藏错误的方法可能会导致访问方法发生变化（例如，访问可信对象的方式可能不同于访问简单对象的方式）。在软件程序中，尤其是基于网络的软件程序中，区分进程或处理器运行缓慢可能颇具挑战。这种区别可能被掩盖，也可能显现出来。

5.2.5 迁移透明性

为了提升性能、可靠性，或掩盖不同主机之间的差异而迁移进程或数据时，用户不应察觉到这些变化。这称为迁移透明性。

5.2.6 性能透明性

性能透明性要求系统的配置不应影响用户对性能的感知。这可能需要使用先进的资源管理系统，也可能需要复杂的资源管理系统。如果资源只能通过性能较低的网络访问，则可能无法做到这一点。

5.2.7 扩展透明性

扩展透明性要求系统能够扩展而不影响应用程序算法。对许多企业而言，系统具备增长和演进的能力至关重要；同时，系统还应能够在必要时缩减规模，并分配所需的空间和/或时间。

5.3 开放性解决方案

要使分布式系统具有开放性，必须公开组件之间清晰且定义明确的接口。这些接口必须标准化，并且新组件应能轻松集成到现有系统中。

5.4 并发解决方案

逐个情况处理：- 这里的简单解决方案是针对这种事件排序添加一条条件语句。如果下一条

消息是链路故障通知，就将该通知存储在内存中，以备之后成为主节点时使用。
复制计算：- 在仅包含一个节点的系统中，存在一个统一的全局事件序列，不会发生任何冲突。

让事件处理程序具备事务性：- 通过让事件处理程序具备事务性来增强它们。事务使一组操作看起来像是同时执行，或者完全没有执行，从而提供一种强大且影响深远的能力。

以无人察觉的方式重新排序事件：-事实证明，使用一种称为虚拟同步的复制模型，可以取得更好的效果。简而言之，虚拟同步为库提供了三个函数：join ( )，用于加入进程组；register ( )，用于注册事件主机；以及 send ( )，用于向整个进程组发送原子广播消息。让自己成为无状态：-
在数据库中，`真实状态` 存储在底层。在网络中，相同的信息存储在交换机自身的路由表中。这意味着控制器所看到的网络状态只是软状态，也就是说，我们始终可以通过查询交换机的当前配置来恢复该状态。

保证自稳定：- 之前的解决方案旨在确保即使其他节点发生故障，系统也始终能正确运行。而这个最终方案是我个人最喜欢的方案，它要乐观得多。1. 确保事务之间相互隔离。
2. 通过以能够保持一致性的方式执行事务，确保数据库一致性。3. 处理读写冲突和写读冲突。

5.5 安全解决方案

计算机和网络地址的三项要求

- 机密性：要求数据仅对授权人员开放访问。
- 完整性：要求只有获授权的团队才能修改数据。
- 可用性：要求授权群组能够访问数据。

5.6 可扩展性解决方案

以下是提高分布式系统可扩展性的方法

- 垂直扩展（增强性能并提升速度）
- 垂直分区（划分工作）
- 水平扩展——应用服务器（委派任务并实现负载均衡）
- 缓存（快速响应）
- 水平扩展与复制 - 数据库服务器（创建副本）

5.7 故障处理方案

容错分布式系统通常分两步处理故障：首先检测故障，其次采取某种恢复操作。检测故障的一种常见方法是使用端到端超时，但使用超时会带来问题。

第六. 参考文献

[1] G. Couloris、J. Dollimore 和 T. Kinberg，《分布式系统–概念与设计》，第 4 版，Addison-Wesley，培生教育，英国，2001 年。*[2]* Andrew S. Tanenbaum 和 Maarten van Steen，2006 年。《分布式系统：原理与范式》（第 2 版）。Prentice-Hall, Inc.，美国。*[3]* Schafer、Dominik、Edinger、Janick、VanSyckel、

分

a、Rajkumar。（2006）。

保护（PPDS）协议：安全分析”。2017

el

”，

国际会议，2014 年，第 73-78 页，DOI:

A. Aloui、M. Msahli、T.

协

第1885-1890页，DOI:

5

CDCSW.2015.30。*[11]* Ghosh、

布式

Ion-Alexandru。

与考量。《国际互联网

）。分布式系统：基本设计挑