南京大學
NANJING UNIVERSITY

Software Quality Attributes and Architectural Patterns

学　　院	:	软件学院
创建时间	:	2023年5月2日
授课教师	:	张贺、潘敏学
姓　　名	:	郁博文
学　　号	:	201250070

1. Architecture Pattern Analysis

1.1 Availability

对于经纪人模式而言，其可能对可用性产生正面影响。因为经纪人可以提供可靠的中介服务，分摊客户端和服务器之间的通信负担，从而减少了单个服务器的负载。同时，经纪人还可以维护多个服务器的状态信息，确保在某些服务器出现故障时，能够快速地进行故障转移，从而保证系统的可用性。

然而，对于点对点模式而言，其可能对可用性产生负面影响。因为在点对点模式中，每个节点都要承担一定的通信负担，而且当节点之间的连接出现问题时，可能会导致整个系统的可用性下降。例如，在我的Assignment 1中，我选择了使用点对点模式实现分布式数据库，但在进行测试时发现，当某个节点离线时，整个系统的可用性会受到影响，导致部分数据无法正常访问。

针对这些可能的负面影响，可以考虑采取以下设计决策来缓解问题：

- 对于经纪人模式，可以采用以下策略来提高可用性：
    1. 采用负载均衡策略，使得客户端请求可以分摊到多个服务器上，从而避免单点故障。
    2. 采用热备份策略，即将主服务器的状态信息实时备份到备用服务器上，当主服务器出现故障时，可以快速地将备用服务器切换为主服务器，从而实现高可用性。
- 对于点对点模式，可以采用以下策略来提高可用性：
    1. 采用冗余备份策略，即将数据同时存储在多个节点上，当某个节点出现故障时，可以从其他节点上恢复数据，从而保证系统的可用性。
    2. 采用节点监控策略，实时监控节点的状态，当某个节点出现故障时，可以快速地检测到并进行故障转移，从而减少节点故障对系统可用性的影响。

1.2 Interoperability

就经纪人模式而言，其可能对互操作性产生正面影响。因为经纪人作为中介服务可以处理不同系统之间的通信，从而提供了一个标准的接口，使得不同系统之间可以进行有效的协作。例如，在一个服务导向的架构中，经纪人可以提供一个标准的消息格式和协议，使得不同服务之间可以通过经纪人进行通信，从而提高了系统的互操作性。

然而，对于点对点模式而言，其可能对互操作性产生负面影响。因为在点对点模式中，每个节点都要承担一定的通信负担，而且不同节点之间的通信可能采用不同的协议和格式，导致不同节点之间难以进行有效的协作。

针对这些可能的负面影响，可以考虑采取以下设计决策来缓解问题：

- 对于经纪人模式，可以采用以下策略来提高互操作性：
    1. 采用标准化协议和格式，使得经纪人能够处理不同系统之间的通信，并将消息转换为标准格式，从而提高不同系统之间的互操作性。
    2. 提供统一的API接口，使得不同系统可以通过该接口进行通信，并对接口进行版本管理，从而实现接口的兼容性和可扩展性。
- 对于点对点模式，可以采用以下策略来提高互操作性：
    1. 统一通信协议和格式，使得不同节点之间可以进行有效的通信，同时确保节点之间的通信遵循相同的规范，从而提高系统的互操作性。
    2. 提供分布式注册表，记录不同节点之间的通信信息和服务信息，从而使得不同节点之间可以进行自动发现和注册，并且保持最新状态，从而提高系统的可扩展性和互操作性。

1.3 Modifiability

就经纪人模式而言，其可能对可修改性产生负面影响。因为经纪人作为中介服务需要处理不同系统之间的通信，从而引入了额外的中间件层，这增加了系统的复杂性，使得系统的修改变得更加困难。例如，在一个服务导向的架构中，修改经纪人可能需要对所有服务进行修改和测试，这增加了系统修改的风险和代价。

对于点对点模式而言，其可能对可修改性产生正面影响。因为在点对点模式中，每个节点都有其独立的功能，节点之间相互独立，这使得系统更加松散耦合，更易于修改和扩展。例如，在一个分布式系统中，可以通过添加新的节点来扩展系统的功能，而不需要修改已有的节点。

针对这些可能的负面影响，可以考虑采取以下设计决策来缓解问题：

- 对于经纪人模式，可以采用以下策略来提高可修改性：
    1. 采用松散耦合的设计原则，使得经纪人服务和其他服务之间的依赖关系尽可能少，从而降低经纪人服务对系统的修改风险。
    2. 采用接口抽象的方式，将经纪人服务与其他服务之间的通信接口分离出来，从而使得修改经纪人服务不会影响其他服务。
- 对于点对点模式，可以采用以下策略来提高可修改性：
    1. 采用模块化的设计原则，将每个节点看作一个独立的模块，从而使得每个节点的修改和扩展都可以独立进行，而不需要影响其他节点。
    2. 采用分布式配置管理工具，对节点的配置信息进行管理和维护，从而方便对节点的修改和扩展，并且保证节点的一致性和可靠性。

1.4 Performance

对于经纪人模式而言，其可能对性能产生负面影响。因为经纪人作为中介服务需要处理不同系统之间的通信，从而引入了额外的中间件层，这会增加通信的延迟和开销。例如，在一个基于REST的服务架构中，通过经纪人来调用其他服务会增加网络通信的次数和额外的HTTP请求，从而导致系统的性能下降。

对于点对点模式而言，其可能对性能产生正面影响。因为在点对点模式中，节点之间直接进行通信，没有中间件层的干扰，这可以降低通信的延迟和开销。例如，在一个对等网络中，节点之间可以通过直接交换消息来进行通信，这可以减少消息的传输次数和额外的通信开销，从而提高系统的性能。

为了解决可能出现的性能问题，可以采取以下设计策略：

- 对于经纪人模式，可以采用以下策略来提高性能：
    1. 使用高效的通信协议，例如gRPC等，以减少通信延迟和开销。
    2. 缓存经纪人服务的响应结果，以减少对其他服务的频繁调用，从而提高系统的性能。
- 对于点对点模式，可以采用以下策略来提高性能：
    1. 使用高效的消息传输协议，例如UDP等，以减少通信延迟和开销。
    2. 使用消息队列等机制，对节点之间的消息进行缓存和处理，以提高系统的并发处理能力和吞吐量。

1.5 Security

对于经纪人模式而言，其可能对安全性产生负面影响。因为经纪人作为中介服务需要处理不同系统之间的通信，从而成为潜在的攻击目标。例如，黑客可能会利用经纪人中间件进行中间人攻击，窃取敏感信息或者对通信进行篡改。另外，经纪人可能需要存储大量敏感信息，因此需要保证数据的机密性和完整性。

对于点对点模式而言，其可能对安全性产生正面影响。因为节点之间直接进行通信，不需要经过中间件，这可以降低攻击者进行攻击的机会。例如，在一个点对点的区块链网络中，节点之间通过加密技术来保护交易数据的安全性，从而保证整个系统的安全性。

为了解决可能出现的安全问题，可以采取以下设计策略：

- 对于经纪人模式，可以采用以下策略来提高安全性：
    1. 使用TLS/SSL等加密通信协议来保证通信安全性。
    2. 对经纪人中间件进行安全配置，例如限制接入权限、加强身份验证等，以减少攻击风险。
- 对于点对点模式，可以采用以下策略来提高安全性：
    1. 对节点之间的通信进行加密，例如使用SSL/TLS协议，以保证通信的机密性和完整性。
    2. 实现身份验证和访问控制机制，例如使用数字证书、访问控制列表等技术，以防止未经授权的节点访问系统。

1.6 Testability

经纪人模式将系统划分为多个组件，并使用经纪人组件进行通信。这种模式可能会降低测试性，因为经纪人可能会成为系统的瓶颈。如果经纪人组件崩溃，那么整个系统的测试都会受到影响。此外，如果系统中的某个组件具有特定的接口，测试人员可能需要使用模拟对象来模拟经纪人组件以进行测试。

另一方面，点对点模式将系统划分为彼此平等的节点，它们可以相互通信并共享信息。这种模式可能会提高测试性，因为每个节点都可以单独测试。测试人员可以使用模拟节点来模拟网络环境，并测试系统在节点之间通信时的行为。

要提高经纪人和点对点模式的测试性，可以使用以下策略：

1. 对于经纪人模式，可以使用冗余经纪人来提高可用性和容错性。这意味着系统中有多个经纪人组件，如果一个组件崩溃，其他组件可以继续工作，从而确保系统的可用性和可靠性。此外，使用模拟对象可以使测试人员更轻松地测试经纪人组件和其他组件之间的通信。
2. 对于点对点模式，可以使用模拟节点来模拟节点之间的通信。这样可以更轻松地测试节点之间的交互，并确保系统在不同条件下的正常运行。
3. 对于两种模式，可以使用自动化测试来提高测试效率和一致性。自动化测试可以减少手动测试的需要，并确保测试是一致的和可重复的。

1.7 Usability

经纪人模式的主要优点是易于管理和控制，因为经纪人负责管理和控制消息传递。然而，这也会带来一些负面影响。由于消息必须通过经纪人进行传递，因此这种模式可能会导致消息传递的延迟和瓶颈。此外，由于经纪人是单点故障，如果经纪人出现故障，整个系统都可能会受到影响。

相比之下，点对点模式具有更好的可扩展性和更高的灵活性。在点对点模式中，每个节点都可以直接与其他节点进行通信，因此消息不需要通过单个中心节点进行传递，因此可以避免由于单个节点故障而导致的系统故障。但是，点对点模式也存在一些负面影响。例如，由于每个节点都必须维护与其他节点的连接，因此可能需要更多的资源和管理工作。

为了解决这些负面影响，可以采取以下策略：

针对经纪人模式：

- 使用多个经纪人：通过使用多个经纪人来避免单个经纪人的故障，同时也可以提高系统的可扩展性。
- 增加带宽和处理能力：通过增加带宽和处理能力来减少延迟和瓶颈。

针对点对点模式：

- 使用中心节点：使用一个中心节点来管理节点之间的连接，从而减少节点之间的管理工作。
- 使用路由协议：使用路由协议来管理节点之间的连接，从而减少管理工作和资源占用。

1.8 Sustainability

对于经纪人模式，由于消息传递需要经过中央经纪人，因此该模式可能需要更多的能源和资源，这可能会影响可持续性。此外，由于中央经纪人是单点故障，如果中央经纪人出现故障，整个系统可能会出现故障，这也会导致资源的浪费和损失。

相比之下，点对点模式通常具有更好的可持续性。由于消息不需要经过中央经纪人，因此该模式可能需要较少的能源和资源。此外，由于点对点模式通常具有更好的容错性，因此在某些情况下，它可能比经纪人模式更具可持续性。

为了提高经纪人和点对点模式的可持续性，可以采取以下策略：

针对经纪人模式：

- 使用高效的经纪人算法：通过使用高效的经纪人算法来降低能源和资源的消耗。
- 部署多个经纪人：通过部署多个经纪人来降低单点故障的风险，从而提高可持续性。

针对点对点模式：

- 优化连接管理：通过优化节点之间的连接管理，可以降低能源和资源的消耗。
- 使用可重复使用的节点：通过使用可重复使用的节点，可以降低资源消耗，并减少废弃物的产生。

1.9 Flexibility

对于经纪人模式，由于中央经纪人可以充当传输协议和消息过滤器，因此该模式可以提供更多的灵活性。在经纪人模式下，可以轻松地添加新的服务或修改现有服务，而不会对系统产生太大的影响。

相比之下，点对点模式通常较少提供灵活性。由于消息直接传递给接收者，因此在点对点模式下，对系统进行更改可能需要重新编写整个应用程序。

为了提高经纪人和点对点模式的灵活性，可以采取以下策略：

针对经纪人模式：

- 保持经纪人的简单性：为了保持经纪人的灵活性，应该避免在经纪人中添加过多的功能。这样可以确保经纪人仍然易于修改和扩展。
- 使用动态服务发现：通过使用动态服务发现机制，可以轻松地添加和删除服务，从而提高灵活性。

针对点对点模式：

- 使用适当的消息格式：通过使用适当的消息格式，可以确保系统具有更好的可扩展性和灵活性。
- 使用消息队列：通过使用消息队列，可以将消息分离出来，从而提高系统的可扩展性和灵活性。

1.10 Scalability

在经纪人模式下，中央经纪人充当了消息路由的角色，因此系统可以更容易地进行水平扩展。当需要处理更多的请求时，可以添加更多的经纪人节点来实现负载均衡。例如，电子商务网站可以通过增加经纪人节点来处理更多的订单。

相比之下，点对点模式可能更难以实现水平扩展。由于消息直接传递给接收者，因此在点对点模式下，增加更多的接收者可能需要重新设计整个应用程序，以处理更多的请求。

为了提高经纪人和点对点模式的可扩展性，可以采取以下策略：

针对经纪人模式：

- 添加更多的经纪人节点：通过添加更多的经纪人节点，可以将负载分布到不同的节点上，从而实现水平扩展。
- 使用缓存：通过使用缓存，可以减少对后端服务的请求量，从而减轻系统的负载。

针对点对点模式：

- 使用消息队列：通过使用消息队列，可以将消息分离出来，从而减轻应用程序的负载。
- 使用异步处理：通过使用异步处理，可以将处理请求的时间分散到多个处理节点上，从而实现水平扩展。

2. Architecture Patterns vs. Quality Attributes & Tactics

2.1 Availability

Quality Attribute	Availability						
Strategies	容错性设计		监测和响应		安全性设计		
Tactics	备份和恢复机制	自动化检测和恢复	实时监测	异常日志记录	预测性分析	访问控制	数据加密
Broker			X	X		X	X
Client Server	X	X	X	X		X	X
Distribute-Cluster	X	X		X		X	X
Event-Driven	X	X	X	X			

Broker模式

- 好处：Broker模式可以将客户端和服务器之间的通信进行解耦，将通信逻辑抽象为中间件层，从而提高可用性。当某个客户端或服务器出现故障时，Broker可以自动进行故障转移，将请求转发到其他可用的服务器上。
- 坏处：Broker模式需要引入一个额外的中间件层，增加了系统的复杂度和延迟，也可能成为系统的单点故障，导致系统的可用性降低。

Client-Server模式

- 好处：Client-Server模式将系统分为客户端和服务器端两个部分，可以将系统的负载分布到多台服务器上，从而提高系统的可用性。客户端和服务器端之间的通信可以通过各种可靠的协议进行，从而保证通信的可靠性。
- 坏处：Client-Server模式需要额外的网络通信开销，可能会影响系统的性能，同时可能存在单点故障问题，导致整个系统的可用性下降。

Distributed Cluster模式

- 好处：Distributed Cluster模式将系统部署在多台服务器上，可以将系统的负载分布到多个节点上，从而提高系统的可用性。当某个节点出现故障时，其他节点可以自动接管该节点的工作，从而保证系统的可用性。
- 坏处：Distributed Cluster模式需要管理和维护多个节点，增加了系统的复杂度和管理难度。节点之间的通信需要使用可靠的协议，否则可能会出现数据不一致等问题，从而影响系统的可用性。

Event-driven模式

- 好处：Event-driven模式将系统的各个组件通过事件进行解耦，从而提高系统的可用性和灵活性。当某个组件出现故障时，其他组件可以继续进行工作，从而保证系统的可用性。

- 坏处：Event-driven模式需要对事件进行合理的设计和管理，否则可能会导致事件过多或过少，从而影响系统的性能和可用性。事件的处理可能需要花费一定的时间，从而影响系统的实时性和响应性。

2.2 Performance

Quality Attribute	Performance						
Strategies	优化算法和数据结构		优化代码		优化系统架构		
Tactics	分割和并行化	选择合适的算法和数据结构	减少重复计算和IO操作	代码并行化和异步化	微服务架构	负载均衡	分布式架构
Broker		X	X				
Client Server		X	X		X		
Distribute-Cluster	X	X	X	X	X	X	X
Event-Driven	X	X	X			X	

Broker模式

- 好处：Broker模式可以将客户端和服务器之间的通信进行解耦，将通信逻辑抽象为中间件层，从而提高系统的性能。中间件层可以对消息进行缓存、聚合、压缩等优化，从而减少通信开销，提高系统的吞吐量。
- 坏处：Broker模式需要引入一个额外的中间件层，增加了系统的复杂度和延迟，也可能成为系统的瓶颈，导致系统的性能下降。

Client-Server模式

- 好处：Client-Server模式将系统分为客户端和服务器端两个部分，可以将系统的负载分布到多台服务器上，从而提高系统的性能。服务器可以使用各种高效的算法和数据结构来处理请求，从而提高系统的吞吐量。
- 坏处：Client-Server模式需要额外的网络通信开销，可能会影响系统的性能，同时可能存在服务器的瓶颈问题，导致整个系统的性能下降。

Distributed Cluster模式

- 好处：Distributed Cluster模式将系统部署在多台服务器上，可以将系统的负载分布到多个节点上，从而提高系统的性能。当某个节点出现故障时，其他节点可以自动接管该节点的工作，从而保证系统的可用性和性能。
- 坏处：Distributed Cluster模式需要管理和维护多个节点，增加了系统的复杂度和管理难度。节点之间的通信需要使用高效的协议和算法，否则可能会出现数据不一致等问题，从而影响系统的性能。

Event-driven模式

- 好处：Event-driven模式将系统的各个组件通过事件进行解耦，从而提高系统的性能和灵活性。组件可以异步地处理事件，从而提高系统的吞吐量和响应速度。
- 坏处：Event-driven模式需要对事件进行合理的设计和管理，否则可能会导致事件过多或过少，从而影响系统的性能和可靠性。事件的处理可能需要花费一定的时间，从而影响系统的实时性和响应性。

2.3 Security

Quality Attribute	Security						
Strategies	访问控制		漏洞管理和修复		日志管理和审计		
Tactics	身份验证和授权	角色和权限管理	漏洞扫描和管理	漏洞修复计划	日志收集和存储	日志分析和监测	审计记录
Broker	X	X					
Client Server	X	X	X				
Distribute-Cluster					X	X	X
Event-Driven			X	X	X	X	X

Broker模式

- 好处：Broker模式可以将客户端和服务器之间的通信进行解耦，通过对中间件层进行安全加固，例如对消息进行加密和数字签名，从而提高系统的安全性。同时，Broker模式可以通过对中间件层进行访问控制，限制不可信的客户端的访问，从而提高系统的安全性。
- 坏处：中间件层可能成为系统的安全漏洞，攻击者可以通过攻击中间件层获取敏感信息或者篡改消息，从而危及系统的安全。

Client-Server模式

- 好处：Client-Server模式可以对服务器进行安全加固，例如限制不可信的客户端的访问，使用安全协议进行数据传输，从而提高系统的安全性。服务器可以使用各种高效的加密算法和安全协议来保护敏感数据，从而提高系统的安全性。
- 坏处：如果服务器被攻击成功，攻击者可以获取敏感信息或者篡改数据，从而危及系统的安全。

Distributed Cluster模式

- 好处：Distributed Cluster模式可以将系统部署在多台服务器上，可以通过增加冗余节点来提高系统的可用性和安全性。当某个节点受到攻击时，其他节点可以自动接管该节点的工作，从而保证系统的可用性和安全性。
- 坏处：分布式系统的管理和维护非常复杂，需要对节点之间的通信和数据传输进行加密和访问控制，否则可能会出现数据泄露或者篡改问题，从而危及系统的安全。

Event-driven模式

- 好处：Event-driven模式可以将系统的各个组件通过事件进行解耦，从而提高系统的安全性。通过对事件进行访问控制和加密，可以限制不可信的组件的访问和篡改，从而提高系统的安全性。
- 坏处：Event-driven模式需要对事件进行合理的设计和管理，否则可能会导致事件过多或过少，从而影响系统的安全性。同时，事件的处理可能会引入新的安全问题，例如拒绝服务攻击或者内存泄漏，从而危及系统的安全。

2.4 Sustainability

Quality Attribute	Sustainability						
Strategies	可靠性和健壮性		资源利用率和优化		技术更新和升级		
Tactics	测试和验证	备份和恢复	资源利用率监测和管理	智能节能	技术评估和规划	技术更新和升级	技术演进和优化
Broker	X	X	X				X
Client Server	X	X	X			X	
Distribute-Cluster	X	X			X		X
Event-Driven	X	X			X	X	

Broker模式

- 好处：Broker模式可以将系统的各个组件通过中间件进行解耦，从而使得系统更加灵活和可扩展，容易进行维护和升级，从而提高系统的可持续性。
- 坏处：中间件层的维护和运营需要耗费一定的资源和成本，从而可能影响系统的可持续性。同时，中间件的运营和维护可能需要耗费大量的能源和环境资源，从而可能影响系统的可持续性。

Client-Server模式

- 好处：Client-Server模式可以对系统的各个组件进行分层，从而使得系统更加清晰和可维护，容易进行升级和扩展，从而提高系统的可持续性。此外，服务器可以利用高效的算法和技术来优化资源利用，从而减少能源和环境资源的消耗，从而提高系统的可持续性。
- 坏处：系统的维护和运营需要耗费一定的资源和成本，从而可能影响系统的可持续性。同时，如果服务器的性能不足，可能会引起系统的性能瓶颈，从而影响系统的可持续性。

Distributed Cluster模式

- 好处：Distributed Cluster模式可以将系统部署在多台服务器上，通过增加冗余节点来提高系统的可用性和可持续性。如果某个节点出现故障或者需要进行升级，可以将其从集群中剔除，从而不影响系统的正常运行。此外，分布式系统可以通过对资源进行合理的分配和利用来优化能源和环境资源的利用，从而提高系统的可持续性。
- 坏处：分布式系统的管理和维护非常复杂，需要耗费大量的人力和物力资源，从而可能影响系统的可持续性。同时，分布式系统的部署和运营需要耗费大量的能源和环境资源，从而可能影响系统的可持续性。

Event-driven模式

- 好处：Event-driven模式可以通过事件进行解耦，从而使得系统更加灵活和可扩展，容易进行维护和升级，从而提高系统的可持续性。此外，事件驱动的系统可以根据实际需要进行资源调度和管理，从而优化能源和环境资源的利用，从而提高系统的可持续性。
- 坏处：事件驱动的系统需要对事件进行合理的设计和管理，否则可能会导致事件过