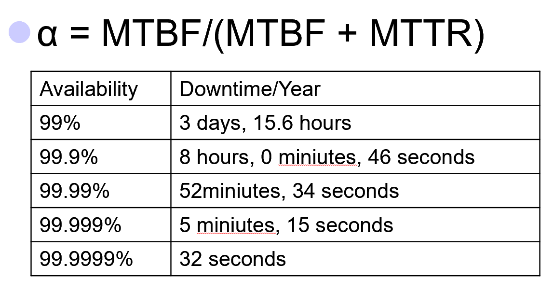
**Chapter 5. Availability**

理解：可用性概念。

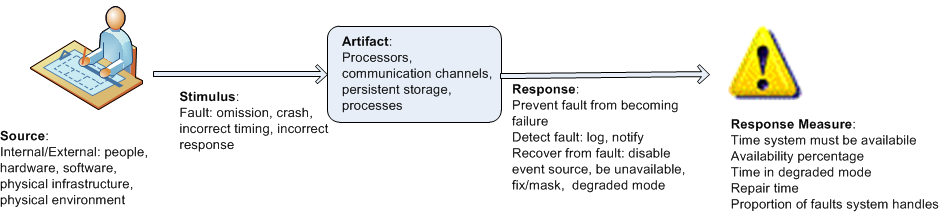
是指软件的一个属性，它已经存在，并且在您需要它的时候已经准备好执行它的任务。广泛的视角，包括通常所说的可靠性。可用性通过添加恢复(修复)的概念建立在可靠性之上。从根本上说，可用性是通过减少故障来最小化服务停机时间。

了解：可用性公式。可用性一般场景。

1. 公式：



1. 一般场景:



1. Source: 内部/外部：人员，硬件，软件，物理基础设施，物理环境
2. Stimulus: 故障：遗漏，崩溃，错误的时间，错误的响应。
3. Artifact：系统的处理器，通信通道，持久存储，进程。
4. Environment：正常运行，启动，关机，修复模式，降级运行，超载运行
5. Response：5.1 防止故障成为故障：

* 检测故障：
* 记录故障
* 通知适当的实体（人员或系统）

5.2 从故障中恢复：

* 禁用导致故障的事件源
* 在修理期间暂时无法使用
* 修复或掩盖故障/故障或包含其造成的损坏
* 在修复过程中以降级模式运行

1. Response Measure:

系统必须可用的时间或时间间隔

可用性百分比（例如99.999％）

检测故障的时间

修复故障的时间

系统处于降级模式的时间或时间间隔

系统防止或处理的某类故障的比例（例如，99％）或速率（例如，高达每秒100次）

1. 场景例子：心跳监视器确定服务器在正常操作期间没有响应。 系统通知操作员并继续操作而无需停机。

掌握：可用性战术。可用性设计清单。

Availability Tactics:

1. Goal: 可用性策略使系统能够承受故障，从而使服务符合其规范。这些策略可以防止故障成为故障，或至少限制故障的影响，并使故障成为可能。



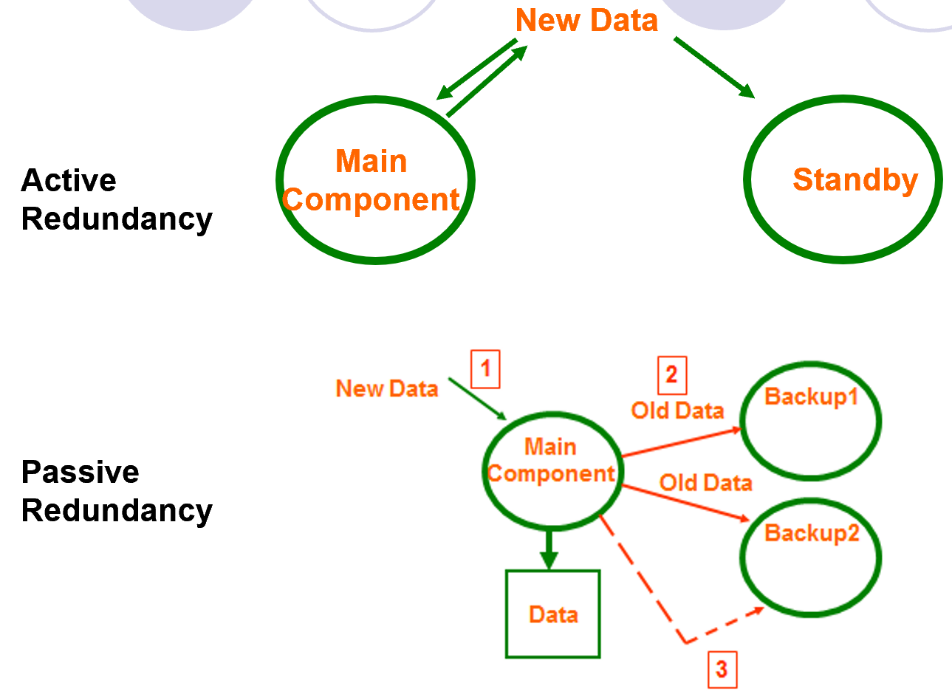
1. 内容



1. **Detect Faults(取决于从各种组件检测生命迹象。)**

* **Ping/echo:** 节点之间交换的异步请求/响应消息对，用于通过关联的网络路径确定可达性和往返延迟。
* **Monitor:** 用于监视系统其他部分的健康状况的组件。 系统监视器可以检测网络或其他共享资源中的故障或拥塞，例如来自拒绝服务攻击。
* **Heartbeat：**系统监视器与被监视进程之间的定期消息交换。
* **Timestamp（时间戳）:** 用于检测不正确的事件序列，主要用于分布式消息传递系统。
* **Sanity(完整性) Checking**: 检查组件操作或输出的有效性或合理性; 通常基于对内部设计的知识，系统的状态或受到详细信息审查的性质。
* **Condition monitoring:** 检查过程或设备中的条件，或验证设计期间的假设。
* **Voting:** 检查重复的组件是否产生相同的结果。
* **Exception Detection:** 检测改变正常执行流程的系统条件。
* **Self-test:** 组件正确操作自行测试的过程。

1. **Recover from Faults (Preparation & Repair:** **正在重试操作或维护冗余数据或计算。)**

* **Active redundancy（冗余）(hot spare):** 保护组中的所有节点并行接收和处理相同的输入，允许redundant spare(s)与active node(s)保持同步状态。
* **Passive redundancy (warm spare):** 只有保护组的active members处理输入流量;为redundant spare(s)提供定期状态更新。
* **Spare(cold spare):**一个保护组的redundant spares保持停止服务。
* **Exception holding:** 通过报告或处理异常来处理异常，可能通过更正异常原因并重试来掩盖错误。
* **Rollback:** **恢复到先前已知的良好状态，称为“rollback line”。**
* **Software upgrade:**以非服务影响的方式对可执行代码进行服务中升级。
* **Retry:** 如果失败是暂时的，则重试该操作可能会导致成功。
* **Ignore faulty behavior:**忽略spurious messages（伪信息）
* **Degradation（降级）:** 维护最关键的系统功能，减少不太重要的功能。
* **Reconfiguration(重新配置):**将责任重新分配给仍在运行的资源，同时保持尽可能多的功能。

1. **Recover from Faults (Reintroduction)：**

* **Shadow(阴影):** 在一段预定义的时间将之前组件恢复为活动角色，在“shadow mode”下运行先前失败或在线升级的组件。
* **State Resynchronization:**与活动冗余和被动冗余相结合，其中状态信息从活动组件发送到备用组件。
* **Escalating**(升级**)Restart**：通过改变重新启动的组件的组成来恢复故障，并最大限度地降低受影响的服务级别。
* **Non-stop Forwarding（转发）**：功能分为监督和数据。 如果监控失效，路由器将继续沿已知路由转发数据包，同时恢复和验证协议信息。

1. **Prevent Faults(取决于从服务中删除元素或限制故障范围。)**

* **Removal From Service:**暂时将系统组件置于服务中断状态，以减轻潜在的系统故障。
* **Transactions：**捆绑状态更新，以便在分布式组件之间交换的异步消息是atomic, consistent, isolated, and durable（原子的，一致的，隔离的和持久的）。
* **Predictive Model:** 监控过程的健康状态，以确保系统在标称参数范围内运行; 在检测到可预测可能的未来故障的条件时采取纠正措施。
* **Exception Prevention:**通过屏蔽故障/智能指针/抽象数据类型/包装器：阻止系统异常来防止系统异常发生。
* **Increase Competence Set:**设计一个组件来处理更多案例 - 故障 - 作为其正常运行的一部分。

1. Availability design:

确定需要高度可用的系统职责。 确保已分配其他职责来检测故障。

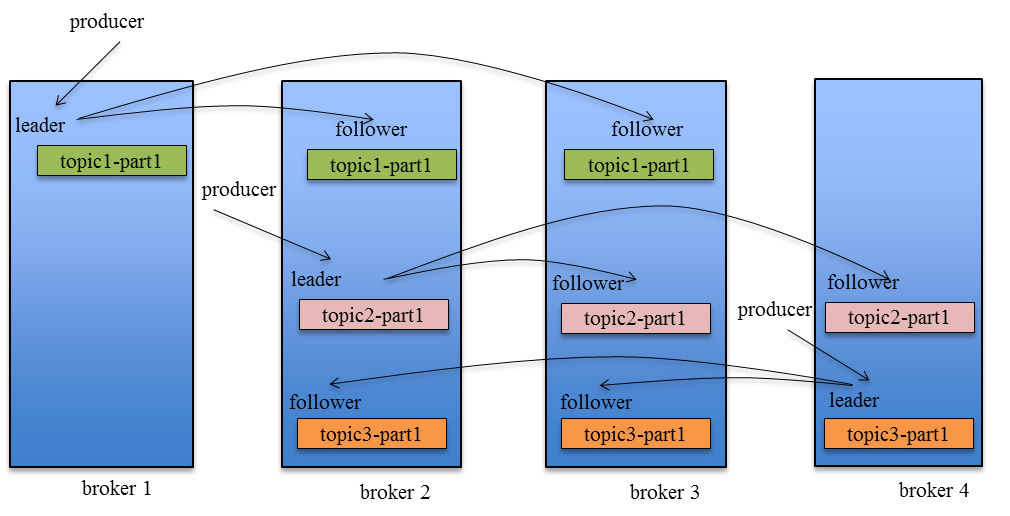
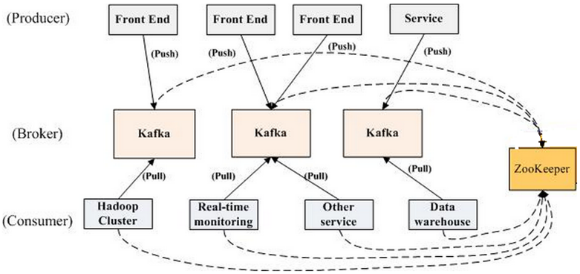
确保有责任：

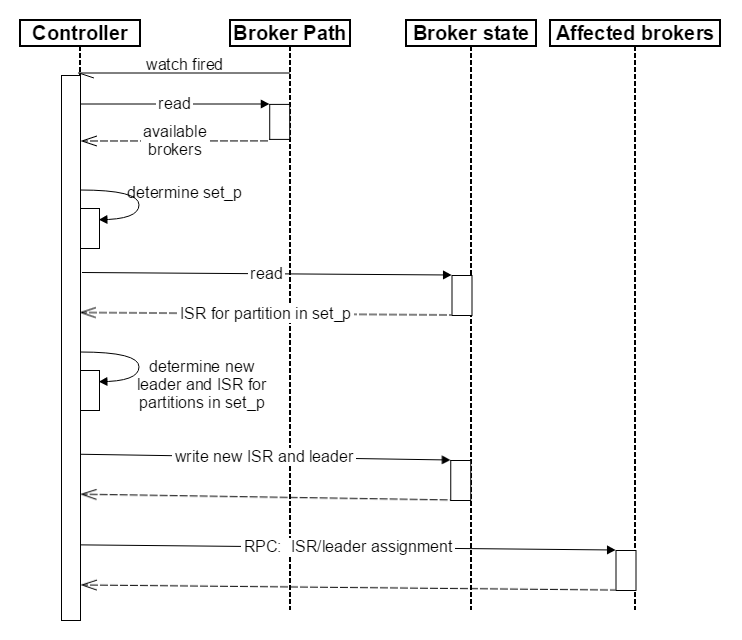
记录故障,通知适当的实体（人员或系统）,禁用导致故障的事件源,暂时不可用,修复或屏蔽故障/故障

以降级模式运行

Study Case（看看就好，知道这个图是availability就好）:

* Kafka High Availability





* StackOverflow 架构

