LEC 4 Primary/Backup Replication

今天的主题是针对容错的主/备份复制,这一想法的极端版本的案例研究。所以我们的主题依然是容错(fault tolerance,也有称 FT):在服务器和网络出现故障时候利用复制提供可用性。

复制可以处理那些类型的故障?

1. 单个副本(single replica)的"错误停止"故障,例如,风扇停止工作,CPU 过热并自动关闭;软件注意到磁盘空间不足并停止运行;有人被副本服务器的电源线或网线绊倒(译者注:意思就是网线或电源线断了?)

2. 可能不是硬件问题也不是软件 BUG,而是人为的配置错误,该错误引发的后果通常不会自动停止,问题可能在服务器间高度相关(i.e.即导致所有副本同时崩溃),但有时这种错误是可以被检测到的(e.g.checksums)。

3. 当发生地震或全市停电时又如何?只有让 replicas 在物理上分离。

单词解析,replica 副本:

由同步复制（Synchronous Replication）产生的，属于一个副本集并且可与该集合中其他副本同步的一个数据库副本。在一个副本中对复制的表的数据所做的更改会发送并应用到其他副本。

注意，英文 replica 和 copy 都有"复制品"的含义，许多文档将它们都翻译为副本，作为术语是有差别的，备份（backup）的数据库副本(copy)与复制(同步复制 replication)的数据库副本(复本 replica)含义不同，前者形成相互独立的副本(copy)，后者形成相关联的副本(复本 replica)，副本之间可以同步。replica 有相同的标识符(ID)，Copy 会有不同的标识符，要进行复制(Replication)，必须要有相同的标识符，即 Replica ID 相同的数据库才能进行复制。

在分布处理等环境下，要将数据的复制品置于异地的服务器上，并使其处于可用状态。这种过程一般称为 Replication 。为了防止出现故障而进行的数据复制称为备份(Backup)而不是 Replication 。

副本值得 Nx 级别的开销吗?

主要有两种生成副本的方法:

1. 状态转移(state transfer),状态转移又可以分为两种,其一为主副本执行服务,其二为主服务器发送[new]state 给备份服务器。

2. 拷贝状态机(Replicated state machine),客户端将操作发送到主服务器,主服务器的操作序列也会发送给备份服务器。所有的副本都会执行所有的操作。如果从相同的状态开始，执行相同的操作、顺序、最终以相同的状态结束。

state transfer 更简单,但是状态可能很大(数据量大),通过网络传输很慢。

Replicated state machine 通常产生更少的网络流量,与“状态”相比,“操作”产生的流量通常很小,但是要做对却很复杂。VM-FT 使用 Replicated state machine,实验 2/3/4 也是如此。

Big Questions:

What state to replicate?(什么状态被复制)

Does primary have to wait for backup?(主服务器需要等待备份服务器吗?)

When to cut over to backup?(什么时候需要切换到备份服务器)

Are anomalies visible at cut-over?(切换时异常是否可见)

How to bring a replacement backup up to speed?(如何提高备份服务器替换的速度)

我们希望 replicas 在什么级别上是幂等的?

应用程序的状态,例如数据库表

GFS 是这样工作

可以是高效的;主服务器仅发送高级别操作给备份服务器

应用程序代码(服务)必须理解容错,例如,转发操作流

机器级别,例如,寄存器和内存内容

可能允许我们拷贝任意现存没有修改的服务器

需要转发机器事件(中断,DMA,等)

需要”机器”修改来发送/接收事件流

今天的论文是(VMware FT) replicates machine-level state

可移植性(Transparent):可以运行在现有的任意操作系统和服务软件上!

对客户端来说就像一台服务器。

概要:

关键词:

hypervisor == monitor == VMM (virtual machine monitor)

操作系统+应用程序是在虚拟机内部运行的“客户机”(O/S+app is the "guest" running inside a virtual machine)

两种机器:主服务器和备份服务器

主服务器通过网络的”日志通道”,并携带日志条目(log entries)发送所有的外部事件(client packets &c)给备份服务器。通常备份服务器的输出被 VM-FT 抛弃。

如果任何一方停止通过网络与另一方通信(无论是 primary 通信不到 backup 还是 backup 通信不到 primary 都可以视为 primary 丢失了 backup),若 primary 上线,它将停止向 backup 发送日志条目。

VMM 会模拟本地磁盘接口，但是实际存储在网络服务器上。

本地磁盘接口被像客户端一样对待:

- 通常只有 primary 和磁盘服务器通信(备份服务器不用管，视为丢弃)
- 如果 backup 上线，它会与磁盘服务器对话

并且外部磁盘能更快的创建新 backup(不用去复制 primary 磁盘)。

primary 何时必须要向 backup 发送信息?

1.在任意可能导致他们的处理结果出现分歧的时候。

2.在执行指令的结果不确定的情况下

FT 必须处理哪些会导致分歧的来源?

1. 只要内存、寄存器相同，在主服务器和备份服务器上执行的指令大部分是相同的，这是通过归纳假设得到的。

2. 外部环境的输入，其实就是网络数据包，这些显示为 DMA 取数据加上中断

3. 中断计时(Timing of interrupts)

4. 不可以是状态函数的指令，例如读取当前时间

5. 不可以用多核处理器，仅用单核处理器

为什么这些分歧会导致灾难?

1.备份上的 b/c 状态将不同于 primary,如果 primary 出现故障，客户端将看到不一致的数据

Ex:GFS 租约到期(GFS lease expiration),

- 假如我们正在复制 GFS 的 master
- chunk server 必须在 60 秒租约到期前发生”please renew”消息
- Clock interrrupt driver master’s notion of time
- 假设 chunk server 会在 60 秒后发生”please renew”消息
- 在 primary 上时钟中断发生在请求到达之后。primary 的主副本将租约更新到同一个 chunk server
- 在备份时，时钟中断刚好在请求之前发生，master 的备份副本将在租约到期后失效。
- 如果 primary 出现故障，backup 将取而代之，它将认为没有租约，并将其授予不同的 chunk server。那么两个 chunk server 将对同一个 chunk 拥有租用权

(时钟中断:是一种硬中断，由时间硬件（系统定时器，一种可编程硬件）产生，CPU 处理后交由时间中断处理程序来完成更新系统时间、执行周期性任务等。linux 时间中断处理程序分种两部分：体系结构相关部分与体系结构无关部

分。体系结构相关部分被注册到内核中，确保中断产生时能执行，这部分不能有耗时操作，主要是更新时间与调用结构无关部分列程（异步）。已到期的定时器由体系结构无关部分来处理，其它的一些耗时操作，如显示时间的更新也在这一部分。)

因此:备份必须能够在指令流中的同一点以相同的顺序看到相同的事件。

每个日志条目:instruction #, type, data

FT 对时间中断的处理

目标：primary 和 backup 在指令流中的同一点看到中断

Primary FT:

FT 接收到 timer interrupt

FT 从 CPU 中读取指令数

FT 在日志通道上发送"timer interrupt at instruction X"

FT 把中断传递给 primary,并恢复它

(这依赖于第 X 条指令后 CPU 对于中断的支持)

Backup FT:

忽略自己的定时器硬件

FT 在 backup 取得指令 X 之前看到日志条目

FT 告诉 CPU 在指令 X 处中断

FT 模拟定时器中断进行备份

注意：备份必须滞后一个日志条目

假设 primary 在指令 X 之后得到一个中断或者输入，如果 backup 已经完成了指令 X，则不能正确的处理输入。所以备份 FT 在看到第一个日志条目之前是无法执行的。

它只执行日志条目的指令，并且在恢复备份之前等待下一个日志条目。

non-deterministic instructions

即使 primary/backup 具有相同的状态，一些指令也会产生不同的结果。例如读取当前时间或周期计数或处理器序列号。

Primary:

如果主处理器执行这样的指令，FT 设置 CPU 中断

FT 执行指令并记录结果

将结果和指令号发送给 backup

Backup:

FT 读取日志条目，并在指令号#设置中断

FT 将提供 primary 获得的值

输出是什么?(发送网络数据包)

primary 和 backup 都执行输出指令，primary FT 实际上做了输出，backup FT 实际上丢弃了输出

QA:

问:如果主服务器在输出之前收到备份确认后立即崩溃,这是否意味着永远不会产生输出？

答:以下是当 primary 出现故障并且 backup 上线时会发生的情况：

- backup 从 primary 获得了一些日志条目。
- 备份继续执行那些日志条目，并丢弃输出。
- 在最后一个日志条目之后，backup 上线并停止丢弃输出。

在我们的示例中，最后一个日志条目是客户端请求的到达，因此在客户端请求到达后，客户端将开始发出输出并且它将向客户端发出回复。

问:但是如果 primary 在发出输出后崩溃了呢？backup 会每秒发出一次输出吗？

答:对 TCP 来说没问题，因为接收方忽略重复的序列号。对写入磁盘来说也没问题，因为 backup 会将相同的数据写入相同的数据块#。

切换时的重复输出在复制系统中非常常见,客户端需要保持足够的状态来忽略重复项或者设计成重复项对系统是无障碍的。

问:FT 能应对网络分区吗?会出现脑裂的情况吗?例如，若 primary 和 backup 都认为另一个已经停机，他们两个会都上线吗?

答:磁盘服务器可以解决这个问题。磁盘服务器支持原子测试和设置(atomic test-and-set)。如果 primary 或 backup 认为其他服务器(其实就是对方)已经死亡，则尝试进行测试和设置(test-and-set)。如果只有一个还活着，它将赢得测试和设置，并开始上线。如果双方都尝试，则有一方将失败并停止。

磁盘服务器可能出现单点故障，如果磁盘服务器 down，服务也将 down(解决方案是复制一个磁盘服务器?)

什么时候 FT 具有吸引力(什么场景下 FT 会被使用)?

1. 关键但低强度的服务，例如 命名服务器(name server)
2. 不常修改的服务

高吞吐量服务的复制是怎样的?

人们使用应用程序级的复制状态机，例如数据库。状态只是数据库，不是所有的内存+磁盘。事件是数据库命令(放或取)，而不是数据包和中断。

结果:更少的细粒度同步，更少的开销。 GFS 使用应用程序级复制，实验 2 和实验 3 也是如此