分布式系统作业

第 1 次作业

姓名：郝裕玮

班级：计科 1 班

学号：18329015

1、分布式系统的扩展方式有哪些？各有哪些利弊？

答：扩展方式共有 3 种：①隐藏通信等待时间；②分布技术；③复制技术。

（1）对于隐藏通信等待时间：

利：①适用于地域扩展。

②使用的是异步通信，使得程序可以尽量避免等待远程服务对请求的响应。

弊：有许多应用程序无法充分利用异步通信的优点。例如,在交互式应用程序中,如果用户发送了一个请求,他一般只会无所事事地等待着回答。

（2）对于分布技术：

利：减轻了单个计算机或服务器的工作量，使得扩大计算和服务规模更加容易。

弊：架构复杂，运维难度增加。

（3）对于复制技术：

利：考虑到可扩展性的问题通常表现为性能的下降,对组件进行复制并将副本分布到系统各处通常是一个好办法。复制不仅能够增加可用性,而且还有助于组件间的负载平衡,从而使性能得到提高。同样,对于在地域上比较分散的系统来说,在请求者附近有一份副本可以在很大程度上隐藏前面提到的通信等待时间的问题。

弊：复制可能会对可扩展性造成不良影响。对数据或任务进行复制后,由于资源存在多个副本,修改其中的一个时会导致它与其他副本不相同,从而导致一致性方面的问题。

2、分布式系统设计存在多个谬误，请针对不同的谬误给出一些具体的技术修正方案。比如考虑到网络是不可靠的，进行请求重新连接等。

答：共有 8 个谬误：

（1）网络可靠：实际上网络是不可靠的，修正方案为：自动重试，进行请求重新连接。

（2）延迟是零：实际上网络调用不是实时的，修正方案为：确保恢复或存储可能需要的所有数据。如将 PB 级别的事务数据存储在云端，并以极高速度将本地与云环境作为同一整体进行操作。

（3）传输带宽是无限的：实际上带宽是有限的，修正方案为：利用数据复制技术同时运行多套数据库，从而利用分布方式进行请求分流。另外，管理员亦能够为特定文件及目录设定更高优先级，确保其在网络当中拥有得到优先传输的权利。

（4）网络是安全的：实际上网络是不安全的，修正方案为：假设网络是敌对的，并不断完善和增强自己的防火墙。

（5）拓扑结构不会改变：实际上为了与不断变化的各种业务保持同步，网络拓扑是在不断变化的。修正方案为：抽象网络的物理结构，构建一个可以处理拓扑变化的，更加灵活的系统。

（6）只有一个管理员：实际上没有某个管理员知道一切，除非系统完全存在于某个小型局域网中，否则将有不同的管理员与网络的各种组件相关联。并且他们将拥有不同程度的专业知识，不同的职责和优先事项。修正方案为：隔离第三方依赖关系，使每个人都要对系统承担起一定的责任。

（7）传输成本为零：实际上传输需要消耗各种资源如 CPU 资源，物理层面上的电缆等传输资源。修正方案为：在开发系统或建立架构时需要将成本高低纳入综合考量。

（8）网络是同质的：同质网络指使用类似配置和相同通信协议的计算机网络。但实际上每个设备具有不同的操作系统，不同的数据传输协议，并且所有设备都与来自各种供应商的网络组件相连。修正方案为：选择标准格式的配置和通信协议以避免供应商锁定，并确保系统间的组件可以相互通信。

3、请从一些开源分布式软件中找出能够体现透明性的样例代码，并解释是何种类型的透明性。

答：寻找的软件为 FastDFS：它是国人开发的一款分布式文件系统，目前社区比较活跃。系统中存在三种节点：Client、Tracker、Storage，在底层存储上通过逻

辑的分组概念，使得通过在同组内配置多个 Storage，从而实现软 RAID10,提升并发 IO 的性能、简单负载均衡及数据的冗余备份；同时通过线性的添加新的逻辑存储组，从容实现存储容量的线性扩容。

选择代码如下所示：

```
static int storage_cmp_by_ip_and_port(const void *p1, const void *p2)
{
    int res;

    res = strcmp(((ConnectionInfo *)p1)->ip_addr, \
            ((ConnectionInfo *)p2)->ip_addr);
    if (res != 0)
    {
        return res;
    }

    return ((ConnectionInfo *)p1)->port - \
            ((ConnectionInfo *)p2)->port;
}

static void insert_into_sorted_servers(TrackerServerGroup *pTrackerGrou
p, \
        ConnectionInfo *pInsertedServer)//该函数体现了访问，位置，复制透明
性
{
    ConnectionInfo *pDestServer;
    for (pDestServer=pTrackerGroup->servers+pTrackerGroup->server_count
; \
        pDestServer>pTrackerGroup->servers; pDestServer--)//访问，位置
    {
        if (storage_cmp_by_ip_and_port(pInsertedServer, \
            pDestServer-1) > 0)
        {
            memcpy(pDestServer, pInsertedServer, \
                sizeof(ConnectionInfo));//复制
            return;
        }

        memcpy(pDestServer, pDestServer-1, sizeof(ConnectionInfo));
    }

    memcpy(pDestServer, pInsertedServer, sizeof(ConnectionInfo));
}
```

```
static int copy_tracker_servers(TrackerServerGroup *pTrackerGroup, \
        const char *filename, char **ppTrackerServers)//该函数体现了访
问，位置，复制，故障透明性
{
    char **ppSrc;
    char **ppEnd;
    ConnectionInfo destServer;
    char *pSeperator;
    char szHost[128];
    int nHostLen;

    memset(&destServer, 0, sizeof(ConnectionInfo));
    destServer.sock = -1;

    ppEnd = ppTrackerServers + pTrackerGroup->server_count;

    pTrackerGroup->server_count = 0;
    for (ppSrc=ppTrackerServers; ppSrc<ppEnd; ppSrc++)//访问，位置
    {
        if ((pSeperator=strchr(*ppSrc, ':')) == NULL)
        {
            logError("file: "__FILE__", line: %d, " \
                "conf file \"%s\", " \
                "tracker_server \"%s\" is invalid, " \
                "correct format is host:port", \
                __LINE__, filename, *ppSrc);//故障
            return EINVAL;
        }

        nHostLen = pSeperator - (*ppSrc);
        if (nHostLen >= sizeof(szHost))
        {
            nHostLen = sizeof(szHost) - 1;
        }
        memcpy(szHost, *ppSrc, nHostLen);//复制
        szHost[nHostLen] = '\0';

        if (getIpaddrByName(szHost, destServer.ip_addr, \
            sizeof(destServer.ip_addr)) == INADDR_NONE)
        {
            logError("file: "__FILE__", line: %d, " \
                "conf file \"%s\", " \
                "host \"%s\" is invalid", \
```

```
                __LINE__, filename, szHost);
            return EINVAL;
        }
        destServer.port = atoi(pSeperator+1);
        if (destServer.port <= 0)
        {
            destServer.port = FDFS_TRACKER_SERVER_DEF_PORT;
        }

        if (bsearch(&destServer, pTrackerGroup->servers, \
            pTrackerGroup->server_count, \
            sizeof(ConnectionInfo), \
            storage_cmp_by_ip_and_port) == NULL)
        {
            insert_into_sorted_servers(pTrackerGroup, &destServer);
            pTrackerGroup->server_count++;
        }
    }
    return 0;
}
```

上述代码体现的透明性有（具体体现位置可见代码注释）：

（1）访问透明性：隐藏数据表示形式的不同以及资源访问方式的不同。

（2）位置透明性：隐藏资源所在位置。

（3）复制透明性：隐藏是否对资源进行复制。

（4）故障透明性：隐藏资源的故障和恢复。