分布式算法

第一章导论

Q1:什么是分布式系统?

> 一个分布式系统式一个能**彼此通信**的单个计算装置的集合(计算单元:硬--处理器,软--进程),包括**紧耦合系统(共享内存)**,**松散系统**(cow,Internet(使用互联网作为通信的协议))

总结: 彼此通信的计算装置集合

Q2:分布式系统与并行处理的区别?

> 具有更高程度的**不确定性**和**行为独立性** 并行处理的目的是使用所有处理器来执行**一个大任务** 分布式系统指每个处理器有自己**独立的任务**,但彼此之间需要协调动作(出于共享资源,可用性,容错等原因) **总结:** 并行处理——**共同任务**,分布式系统——**独立任务**但需要协调

Q3:分布式系统作用?

> - 1.共享资源
> - 2.改善性能:并行求解问题
> - 3.改善可用性,提高可靠性,防止某些成分发生故障

总结: 方便(共享资源),快(改善性能),可靠(防止宕机导致系统不可用)

Q4:分布式系统当前面临的困难?

> - 异质性:软硬件环境,计算介质不同
> - 异步性:时间发生的绝对甚至相对时间不可能总是精确知道
> - 局部性:每个计算实体只有全局情况一个局部视图(无法获取全局信息)

总结: 设备多样性,无绝对时间,无全局状态视图,结点随机故障

Q5:分布式算法研究方法

> - 1.对各种分布式情况发生的**问题进行抽象**
> - 2.精确的**陈述问题**
> - 3.设计和分析有效算法**解决问题**
> - 4.**证明算法的最优性**

总结: 抽象,陈述,设计,证明

Q6:设计分布式算法需要考虑的因素?

> - 通信: 计算实体间通过**msg传递信息**还是使用**共享变量**(本课程只考虑msg传递消息)
> - 可用信息: 哪些**计时信息**,**行为**是可用的(体现异步性以及局部性)
> - 容错: 哪些错误是可以容忍的

总结: 通信,可用信息,容错

Q7:分布式算法中有哪些衡量算法性能的指标

- 1.时间复杂度:从算法第一个结点启动,到最后一个结点结束所需要的时间
- 2.空间复杂度:额外的辅助空间,共享变量大小等
- 3.信息复杂度:在基于msg传递模型中,为了保证算法正确性,所需要传递的信息总数
- 4.故障/非故障节点个数(本课程不考虑)

总结: 时间/空间/msg

Q8:常用的分布式抽象模型有哪些? 本课程主要考虑三种模型,按照**通信介质(共享存储/msg传递)** 以及 **同步程度(同步/异步)** 进行划分

- 异步共享存储模型:紧耦合,处理器时钟信号不来自同一信号源(通信介质:共享内存,同步程度:异步)
- 异步msg传递模型:用于松散耦合及其,以及广域网(通信介质:msg,同步程度:异步)
- 同步msg传递模型:**理想的msg传递系统**,该系统中,计时信息(如msg延迟上界)已知,系统划**分为轮执行,是异步系统的一个特例**.(通信介质:msg,同步程度:同步)

总结: 同步/异步+共享存储/msg,其中同步的共享存储可以直接理解为并行计算因此不考虑

Q9:分布式系统的错误种类?

- **初始死进程:** 局部算法没有执行,直接跳过
- **Crash failure崩溃错误(损毁模型):** 没有警告在某点停止操作
- **拜占庭错误:** 一个出错可能引起任意动作,执行了与局部算法不一致的任意步.该错误发生的进程可能发送的信息包含任意内容(**设计程序时应该尽量避免**)

总结: 跳过(初始死进程),宕机(崩溃损毁),随机(拜占庭)

第二章 消息传递系统中基本算法

前提:本章研究的是无故障的msg传递系统,考虑两种主要的记时模型:同步/异步

Q1:消息传递系统和的形式化模型有哪些基本概念?

- 拓扑结构:无向图,结点代表处理机,边代表**双向**信道
- 算法:由每个处理机上的局部程序构成,**局部程序**包括:
    - 1. 执行局部计算
    - 2. 发送/接受msg
- 状态:由pi的变量,pi的msgs构成,pi的每个状态由2r个msg构成(r为邻居个数)
    - $outbuf_i[l](1 \leq l \leq r)$:表示pi经第l条关联的信道发送给邻居,但还未传到邻居的msg
    - $inbuf_i[l](1 \leq l \leq r)$:表示pi的第l条信道上已传递到pi,但未经pi计算处理的msg
- 初始状态:Qj包含一个特殊的初始状态子集,每个 $inbuf_i[l]$ **为空**,但 $outbuf_i[l]$ **不一定为空**.
- 转化函数:处理器pi的转换函数,实际上是一个局部程序,将输入缓冲区中的信息情况并产生输出
- 配置:配置是分布式系统在某点上整个算法的**全局状态**
- 事件:系统中发生的事情模型化为事件,对于msg传递系统中主要分为两种:
    - omp(i)--计算事件,代表处理器pi的一个计算步骤,其中pi转换函数被用于当前可访问状态
    - del(i,j,m)传递事件,表示msg m从pi传递到pj
- 执行:系统在事件上的行为被模型话为一个执行

- 安全性条件:表示某个性质在**每次执行**中每个**可到达配置**里必须成立(在序列的每个优先前缀里必须成立的条件),从非形式化语言表示:坏事从不发生,**保证运行合法**
- 活跃性条件:表示某个性质在每次执行中的某些可达配置里必须成立,必须成立一定次数的条件,非形式化的语言描述:最终某个好事将会发生,**保证程序可终止**

总结: **拓扑结构**用于描述图链接信息,**算法**为局部程序的集合,局部程序将某个节点的**状态**通过**转化函数**转化为下一个状态,这个过程称之为一个**事件**,对于一个事件进行的局部程序我们称之为一个**执行**,对于所有节点在某事件发生后的全局状态称之为一个**配置**,对于每次执行均成立的性质称之为**安全性条件**(通常情况下我们将该性质定义为执行合法的条件),至少存在一个执行使得某性质成立称之为**活跃性条件**(通常情况下我们需要让程序终止满足该条件)

Q2:如何证明满足安全性? 三种方法

- 1.定义:对于所有$r \in I, P(r)$成立且$P_i \rightarrow P_{i+1}$,断言P总是成立,在每个初始配置中断言P总是成立,并且在每一次转移中保持不变,因此在每个可达配置中不变式成立
- 2.定理一:如果P是S的一个不变式,那么对于S的每次执行的每一配置,P均成立
- 3.定理二:设Q是S的不变式,设$Q \rightarrow P$对于每一个$r \in C$, P在每一执行的每一配置中均成立.
- 其实还有一个一级导出的,太麻烦了就自己看吧

总结: 1.证明**所有**初始状态均成立,且在转移中不变 2.证明为不变式 3.证明存在一个不变式可以推到该结论

Q3:如何证明满足活跃性条件?

- 1.证明有限,即算法不会无限执行下去
- 2.构造一个良基集(不存在无穷递减的偏序序列)
- 3.找到一个范函数,将转移系统的每个配置映射到上述良基集中
- 4.证明不会死锁能正常终止

总结: 这个比较迷,不考的概率比较高,有兴趣可以自己了解

Q4:异步系统有哪些特殊限制/不同?

- msg传递事件与处理器两个相邻步骤之间的**时间无固定上界**.例:e-mail传递时可能需要数天,因此设计异步算法时需要**特殊的计时参数**,不能依赖该上界.
- 执行片段:与上述定义不同,这里的执行片段表示的是指一个配置以及事件交替构成的序列
- 执行:这里的执行要求第一个配置必须为初始配置
- 调度:执行去除所有配置之后的序列,如果局部程序是确定的(无随机),则执行可以由初始配置以及一个调度唯一确定
- 容许执行:对于拥有无限计算事件的异步系统中来说,其容许执行表示每个计算的msg都最终被传递(处理器无出错),即消息不会丢失

总结: 无

Q5:同步系统有哪些特殊限制/不同?

- 同步执行分轮进行执行,**每轮由一个传递事件以及一个计算事件组成**(先进行消息的传递,后进行计算任务处理传递的信息)
- 容许的执行:指无限的执行,因为轮的结构因此每个处理器可以**执行无限数目的计算步**,同时每个被发送的**msg最终被传递**

Q6:同步系统与异步系统的区别?

> - 在一个无错的同步系统中,**一个算法的执行只取决于初始配置(同步系统)**,但在异步系统中对于相同的初始配置以及无错假定,**由于处理器步骤间隔以及消息延迟的不确定性,因此对于相同算法可能由不同的执行.**

总结: 异步算法需要考虑消息传递延迟导致的先后区别

Q7:什么是算法的msg复杂性?

> - 算法在**所有容许执行**上发送的msg总数的最大值(包括同步系统以及异步系统)
> - 衡量标准:
>
> > - 1.消息条数(不考虑消息位数)
> > - 2.消息总位数(如2个8字节的消息,比一个64位的消息更小)

Q8:什么是算法的事件复杂度?

> - **同步系统:** 最大轮数,算法的任何容许执行知道终止的**最大轮数**
> - **异步系统:** 首先我们基于下面两个假设
>
> > - 假设1.节点计算任何有限数目事件的时间为0
> > - 假设2.消息发送和接受之间的时间假设为1
>
> - 那么基于上述两个假设的情况下,异步系统的时间复杂度为所有计时容许执行中到终止的最大时间.

Q9:异步算法中什么是计时执行(timed execution)

> - 每个事件关联一个非负实数,表示事件发生的时间,时间起始于0且非递减(**对于单个处理器时严格增的**)因此执行的时间可以**根据时间进行排序**

总结: 可能存在时间戳相同的事件(因为异步),单处理器上不存在相同事件戳的不同事件