实验大作业:

按照<<一种混合并行XML解析方法>>论文,

1. 设计程序总体基本框架并实现:
2. 将XML文件数据读入作为第0阶段，分块读入缓存，多个缓存块形成缓存链表；
3. 建立多线程并行处理不同数据块的数据结构和程序逻辑控制结构。
4. 实现XML并行解析程序第一阶段即事件划分阶段的功能。
5. 建立<类型和位置的数据结构，将解析到的<类型和位置（在缓存块的相对位置和在XML文件中的绝对位置信息保存到相应数据结构中）。

程序总体思路：

1. XML文件按数据块分块读入内存，设置建立数据块的数据结构（数据块已经完成的阶段，尚未完成的阶段，包含的元素内容及起始，结束位置等），采用动态链表形式。
2. 第一阶段即事件划分阶段采用一个单独线程实现；
3. 第二阶段即事件解析阶段采用多个线程实现，实现基于数据块的多线程并行解析。
4. 第三阶段即后处理阶段采用一个单独线程实现；
5. 各阶段之间实现基于数据块的流水线并行处理。

思路:

1. 建立一个读入线程，该线程建立多个缓冲区的动态链表结构,每个缓冲区是固定长度(比如10k), 把文件数据的剩余数据中每次读固定长度的数据到一个新的缓冲区；每个新的缓冲区建立一个区号；
2. 建立一个解析<线程，对新的缓冲区的<号作解析判断，记录<号的位置和类型；

位置信息包括：离文档开始处的距离；

所处当前缓冲区号，离所处缓冲区开始处的位移量；

类型信息可以定义一个枚举数据类型；

位置信息和类型信息形成一个数据结构体bcs,  
 define BUFLEN 10240

typedef enum Bcstype{

StagorEmptytag\_start,

Etag\_start,

PI\_start,

Content,

CDSECT\_start,

COMMENT\_start

} Bcstype;

Struct bcs\_t{

int fileoffset;

Int bufnum;

int bufpos;

Bcstype bt;

}Bcs;

typedef struct Bcsarray\_t{

Bcs bcs[BUFLEN/10];

struct Bcsarray\_t \*next;

} Bcsarray;

typedef struct buffer\_t{  
 char buf[BUFLEN];

Bcsarray bcsay;

int bufnum;

int FINISH\_STAGE0;

int START\_STAGE0;

int FINISH\_STAGE1;

int FINISH\_STAGE2;

int FINISH\_STAGE3;

int START\_STAGE1;

int START\_STAGE2;

int START\_STAGE3;

Struct buffer\_t \*next;

}databuf;

1. 读入线程和解析<号线程并行运行，同时每个线程开始前都需要检索和修改缓冲区的以下状态信息（FINISH\_STAGE0，START\_STAGE0……）,这些状态信息是竞争资源，需要采用锁机制；

线程运行结束处同样需要修改缓冲区状态（不需要锁，想想为什么？）

1. 第二阶段任务:
2. Name, attribute的起始和结束位置，
3. 所有字符合法性范围的检查；
4. Stag和Etag的匹配性检查（文档结构是否合法），要求Stag和Etag的位置匹配正确且name是一样的。

第二阶段，应该建立一个新的数据结构，存储所有解析到的事件类型和起适结束位置

（stag,stag的name, attribute, attribute的name和attvale, etag, pi, cdsect, comment, pi, Chardata）

1. 第三阶段任务：
2. 主要是检查不在同一块的stag,与etag是否匹配；
3. 在第三阶段可以将第二阶段的各块的输出结果连起来。