uxlock-master 源码说明

目前的阻塞锁在block-exp分支上，需要git switch切换过去

框架和编译

在litl目录下直接 make ，会生成不同锁策略的动态库，在跑benchmark时只需要加载这些动态库就可以直接替换到默认的pthread_mutex_lock策略

如果加入一种新的策略

1. 在src和include目录下加入你的源代码，如我想设计一种新的锁策略名为kkk，就添加kkk.c，并添加相关函数，并在include目录下加入头文件kkk.h， 并且#define LOCK_ALGORITHM "KKK"， 具体可参考已有的策略
2. 在interpose.c中根据LOCK_ALGORITHM 加入头文件
3. 在makefile.config中加入kkk_original，这里的original代表阻塞策略自带不适用litl框架的waiting_policy.h，如果想使用则为kkk_spinlock等等
4. 重新make，会生成libkkk_original.sh
5. 跑程序 bash libkkk_original.sh bench + 参数 就会将默认pthread_mutex_lock替换为kkk策略

Micro-bench 设计

路径：uxlock/liti/bench/bench.c

命令行参数

Usage:

-h 打印参数说明

-t 线程数

-g 长临界区长度

-s 短临界区长度

-S 短临界区线程数

-d 两次拿锁之间的间隔

-l SLO策略下 `target_latency`

-T 测试总时长

-u ux线程数

执行流程

1.初始化参数缺省值

2.接受命令行参数

3.创建参数t个线程，并发执行测试函数，执行时间由参数T控制

4.打印输出各个核的吞吐量与延迟

脚本

目录：uxlock/litl/scripts

功能说明

measure.sh：得到bench输出的吞吐量与时延，先打印各核心的p50,avg,p99,tail latency，throughput，最后打印指定部分核心的总体时延与吞吐率。

如不需要打印各个核心的时延，注释以下代码即可

```
    # echo -n "Core $i Cnt $line P99 "
    # sed -n "$per_core_p99_line p" tail_$i-sorted | tr '\n' ' '
    # echo -n "P95 "
    # sed -n "$per_core_p95_line p" tail_$i-sorted | tr '\n' ' '
    # echo -n "P50 "
    # sed -n "$per_core_p50_line p" tail_$i-sorted | tr '\n' ' '
    # echo -n "avg "
    # awk '{sum += $1} END {printf "%3.3f\t",sum/NR}' tail_$i-sorted | tr '\n' '
    # echo -n "Tail "
    # tail -n 1 tail_$i-sorted
```

run_exp1.sh~result7.sh：各个实验

以下列代码为例：

```
delay=0
core=6
time=1
LONG_CRI=8
SHORT_CRI=8
uxthread=6

echo "uta "
for i in 6
do
        $LITL_DIR/libuta_original.sh  $LITL_DIR/bin/uta_bench -u $uxthread -t $i
        $LOCAL_DIR/measure.sh ./result $i $uxthread 0
done
```

- 1-6行定义命令行参数
- $LITL_DIR/libuta_original.sh 表示当前用uta策略替换默认策略
  - 例子：使用uta.c 中的 `uta_mutex_lock` 去替换测试文件的 `pthread_mutex_lock` 函数
- 11行为使用uta策略跑uta_bench，6个线程，其中有6个ux线程，0个非ux线程，短临界区线程数6个，短临界区大小与长临界区大小均为8，输出存储到result文件中
- 12行表示用measure.sh分析result中的数据，第一个参数为数据存储的文件result，第二个参数i表示有i个线程，第三个参数uxthread表示打印出uxthread个线程的数据

脚本与实验对照关系

脚本在script目录下

- run_test_local.sh 中有mutexee, mutex, proto， mcs策略的测试
  - 利用measure.sh收集数据打出p50, avg, p99 时延，具体逻辑可以看这个文件

三种uta策略

目录：uxlock/litl/src

UTA-SCF: 对应src/uta.c

UTA-SLO: 对应src/utaspc.c

UTA-HTF: 对应src/utafts.c

UTA-SCF策略

宏说明

- SHORT_BATCH_THRESHOLD：执行多少次短临界区执行一次长临界区，该值必须小于2^16 （即65536）
- ADJUST_FREQ：动态调整临界区阈值，该值应小于SHORT_BATCH_THRESHOLD
- DEFAULT_SHORT_THRESHOLD：默认临界区阈值（区分是短临界区还是长临界区）
- NO_UX_MAX_WAIT_TIME：非ux线程最长等待时间

UTA-FTS策略

宏说明

- DEFAULT_REFILL_WINDOW： 默认budget大小，该值大小会影响FTS策略性能，之前的测试为在临界区长度40倍时性能较好

UTA-SLO策略

宏说明

- DEFAULT_SHORT_THRESHOLD：动态阈值，该值过低会导致队列为空长临界区线程频繁进入队列
- DEFAULT_REORDER: 默认reoder_window大小
- DEFAULT_ADJUST_UNIT：每reorder window变动调整大小，过小会导致reorder_window上升过慢失去效果