Homework4: OpenCilk Tutorial

杨浩然 10195501441

操作系统： ubuntu 20.04

2021-10-26

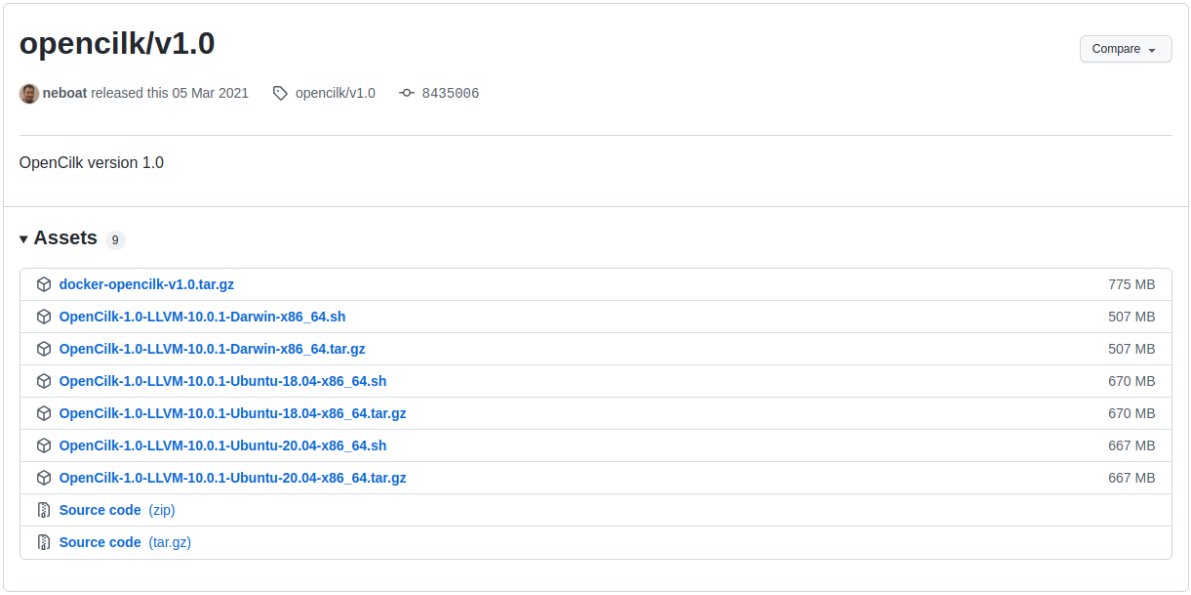
安装 OpenCilk 并记录执行过程，了解并尝试 pre-build binaries 和 build from source code 两种安装方式

Precompiled binaries

在 https://github.com/OpenCilk/opencilk-project/releases/tag/opencilk%2Fv1.0 找到对应系统的 OpenCilk。

这里选择的是 OpenCilk-1.0-LLVM-10.0.1-Ubuntu-20.04-x86_64.tar.gz

如果下载速度太慢可以访问 github 镜像站下载。



解压即可。

Build from source

在 https://github.com/OpenCilk/infrastructure/blob/release/INSTALLING.md 查看安装提示。

- 克隆 infrastructure 仓库

```
git clone -b opencilk/v1.0 https://github.com/OpenCilk/infrastructure
```

```
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5$ git clone -b opencilk/v1.0 https://github.com/OpenCilk/infrastructure
正克隆到 'infrastructure'...
remote: Enumerating objects: 208, done.
remote: Counting objects: 100% (208/208), done.
remote: Compressing objects: 100% (127/127), done.
remote: Total 208 (delta 99), reused 139 (delta 59), pack-reused 0
接收对象中: 100% (208/208), 43.36 KiB | 1.11 MiB/s, 完成.
处理 delta 中: 100% (99/99), 完成.
注意：正在切换到 '7918a23df3201d8721291059ae6841ad586482ca'。

您正处于分离头指针状态。您可以查看、做试验性的修改及提交，并且您可以在切换
回一个分支时，丢弃在此状态下所做的提交而不对分支造成影响。

如果您想要通过创建分支来保留在此状态下所做的提交，您可以通过在 switch 命令
中添加参数 -c 来实现（现在或稍后）。例如：

  git switch -c <新分支名>

或者撤销此操作：

  git switch -

通过将配置变量 advice.detachedHead 设置为 false 来关闭此建议
```

- 运行 get 脚本来获取 OpenCilk 源码

```
infrastructure/tools/get $(pwd)/opencilk
```

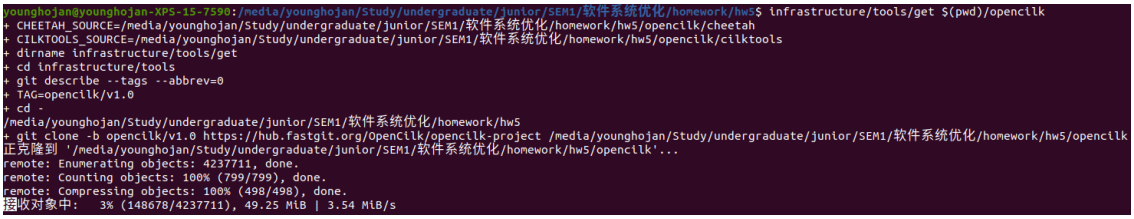
```
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5$ infrastructure/tools/get $(pwd)/opencilk
+ CHEETAH_SOURCE=/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/opencilk/cheetah
+ CILKTOOLS_SOURCE=/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/opencilk/cilktools
+ dirname infrastructure/tools/get
+ cd infrastructure/tools
+ git describe --tags --abbrev=0
+ TAG=opencilk/v1.0
+ cd -
/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5
+ git clone -b opencilk/v1.0 https://github.com/OpenCilk/opencilk-project /media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/opencilk
正克隆到 '/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/opencilk'...
remote: Enumerating objects: 4237711, done.
remote: Counting objects: 100% (799/799), done.
remote: Compressing objects: 100% (498/498), done.
接收对象中:   0% (10574/4237711), 3.90 MiB | 116.00 KiB/s
```

速度极慢，建议先摸一会儿鱼。

等得头皮发麻，直接把脚本里面的 https://github.com/ 改成镜像站 https://hub.fastgit.org/。

```
#!/bin/sh

usage() {
    printf "USAGE:\n"
    printf "%s source-dir\n" "$0"
    printf "\tClone the OpenCilk source into the directory source-dir.\n"
    exit 1
}

# Source directory for OpenCilk (root of the opencilk-project repository tree).
# Must be an absolute pathname.
while [ $# -gt 0 ]
do
    case "$1" in
        /*)
            OPENCILK_SOURCE="$1"
            shift;;
        -h|-help|--help) usage;;
        -*) echo "unknown option $1"; exit 1;;
        *) echo "First argument ($1) must be absolute pathname"; exit 1;;
    esac
done

if test -z "${OPENCILK_SOURCE}" ; then
    usage
fi

set -ex

# Set the directories into which to clone cheetah and productivity-tools.
CHEETAH_SOURCE="${OPENCILK_SOURCE:?}/cheetah"
CILKTOOLS_SOURCE="${OPENCILK_SOURCE:?}/cilktools"

# We shall clone the three OpenCilk source repositories using the same
# tag as this infrastructure repository.
cd "$(dirname "$0" )"
TAG="$(git describe --tags --abbrev=0)"
cd -

# Clone the three OpenCilk source repositories.
git clone -b "${TAG}" https://hub.fastgit.org/OpenCilk/opencilk-project "${OPENCILK_SOURCE}"
git clone -b "${TAG}" https://hub.fastgit.org/OpenCilk/cheetah "${CHEETAH_SOURCE}"
git clone -b "${TAG}" https://hub.fastgit.org/OpenCilk/productivity-tools "${CILKTOOLS_SOURCE}"

# Report that the clone succeeded.
printf "Clone completed successfully.  Any messages above about 'detached HEAD' states are expected behavior.\n"
~
~
~
```

速度明显提升，继续摸一会儿鱼。



- 运行 build 脚本来编译 OpenCilk 源码

```
infrastructure/tools/build $(pwd)/opencilk $(pwd)/build
```

Oops...出错了

貌似是机器内存太小，需要减少并发数。在上面的命令中加上参数，以设定并发数，如

```
infrastructure/tools/build $(pwd)/opencilk $(pwd)/build 8   # 只准用 8 个线程
```

build completed

阅读教程“Cilk Tutorial”，理解 cilk_spawn，cilk_sync，cilk_for 和 locks 等基本实现

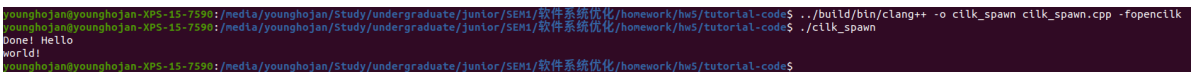
Cilk 是干啥的?

Cilk 是 C/C++ 的 extension，在进行 C/C++ 编程时，可以利用 Cilk 进行多核并行编程。

`cilk_spawn`

对于一个函数，如果在 call 这个函数时，在前面加上 `cilk_spawn`，那么这个函数可能会和 caller 并行执行。

运行 C++ 程序示例看看（示例代码均不附上）：



`cilk_sync`

使用 `cilk_spawn` 可能会让函数乱序运行，使用 `cilk_sync` 可以避免 `cilk_spawn` 的缺点。

写入 `cilk_sync` 会使前面使用了 `cilk_spawn` 的 tasks 互相等待。

运行 C++ 程序示例:

`cilk_for`

`cilk_for` 就是普通 `for` 的并行版本，它把一个 for 循环切成很多个片片，使用不同的线程执行。每个片片可以包含的最大迭代次数成为 **grain size**。grain size 可以用 **cilk grainsize pragma** 定义，比如：

```
#pragma cilk grainsize = min(2048, N / (8*p))   # N 是循环次数，p 是并发的线程数
```

使用 `cilk_for` 时，Intel compiler 会在半途停止每个循环，直到每个循环的迭代次数小于或等于 grain size。

`cilk_for` 的使用有以下的限制：

1. 不能在循环里面改变循环控制变量的值。
2. 不能在循环外部声明循环控制变量。

运行 C++ 程序示例：

```
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/tutorial-code$ ../build/bin/clang++ -o cilk_for cilk_for.cpp -fopencilk
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/tutorial-code$ ./cilk_for
50005000
```

Locks

这个就是使用锁这种同步机制来防止多个线程并发地修改同一变量的值，以此减少多线程数据竞争的发生。

运行 C++ 程序示例：

```
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/tutorial-code$ ../OpenCilk-10.0.1-Linux/bin/clang++ -o locks locks.cpp -fopencilk -ltbb
locks.cpp:3:10: fatal error: 'tbb/mutex.h' file not found
#include <tbb/mutex.h> //mutex library
         ^~~~~~~~~~~~~
1 error generated.
```

这里烦了很久，google 了以下，发现是缺少包，需要进行安装：

```
sudo apt install libtbb-dev libtbb-doc libtbb2
```

安装之后再编译：

```
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/tutorial-code$ ../OpenCilk-10.0.1-Linux/bin/clang++ -o locks locks.cpp -fopencilk -ltbb
locks.cpp:10:19: error: expected '(' for function-style cast or type construction
    cilk_for (int i = 0; i <= 10000; i++){
                  ~~~ ^
1 error generated.
```

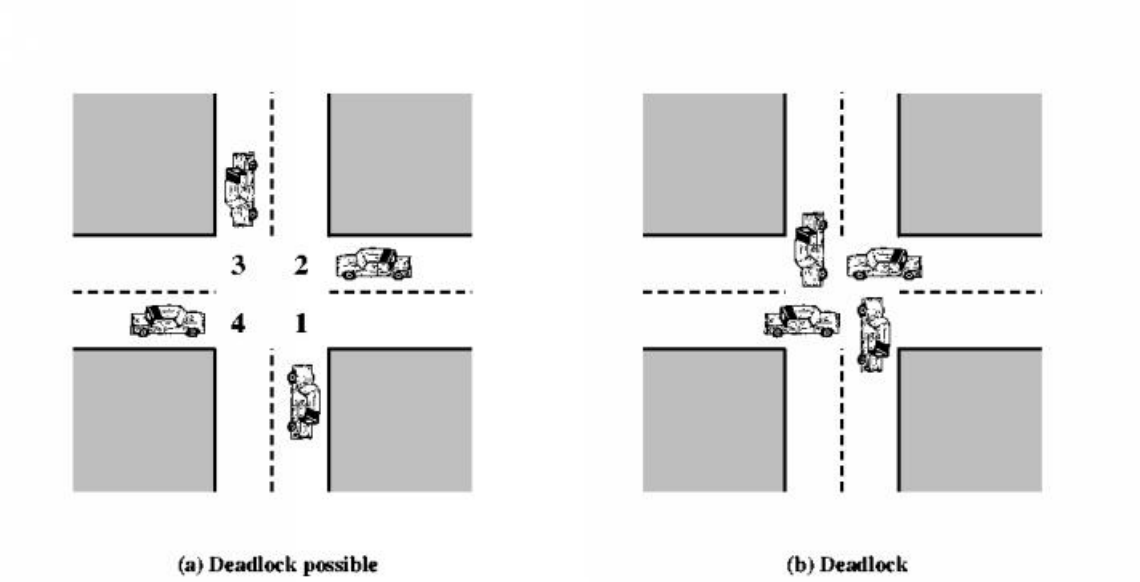
这个奇怪的错误，上学期《操作系统》课程的实践中也遇到过，当时是在编译 MINIX 的源码。

想了一下，这个错误大概率跟 `cilk_for` 相关。发现 tutorial 的代码中，没有 `#include <cilk/cilk.h>`，但之前的 tutorial 代码中均有 `#include <cilk/cilk.h>`（想了一下，也是我蠢，tbb/mutex.h 中不包含 cilk 相关的函数，那么肯定需要把 cilk/cilk.h 包含进来咯，但是为什么这里的错误信息不是未包含 cilk/cilk.h 呢，奇怪），所以在代码中加入 `#include <cilk/cilk.h>` 即可。注意编译时还要加上 `-ltbb` 来启用锁，运行结果如下：

```
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/tutorial-code$ ../OpenCilk-10.0.1-Linux/bin/clang++ -o locks locks.cpp -fopencilk -ltbb
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/tutorial-code$ ./locks
50005000
```

Reducers (C++ Only)

使用 Locks 可以极大减少 data race 的情况，但是死锁还是有可能发生的。



这时更好的选择是 Reducers，**reducer** 是一个可以由多个并行线程安全使用的变量。它的做法是，为每个线程提供一个 private 的变量副本，交由各个线程互不相干地使用，当各线程同步时，各自拥有的副本将被合并。

运行 C++ 程序示例：

```
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/tutorial-code$ ../build/bin/clang++ -o reducers reducers.cpp -fopencilk
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/tutorial-code$ ./reducers
50005000
```

Run Time System Functions

Cilk 还提供了一些函数，使得用户可以在程序运行时，对程序行为进行一些细节上的调整。

1. `int __cilkrts_set_param(const char* name, const char* value);`

   此函数允许用户在程序运行时修改函数的参数，`name` 是需要修改的参数名，`value` 是新的参数值。

2. `int __cilkrts_get_nworkers(void);`

   此函数返回运行时的线程数。

3. `int __cilkrts_get_worker_number (void);`

   此函数返回当前被调用函数被分配的线程数。

4. `int __cilkrts_get_total_workers (void);`

   此函数返回 run time system 分配的所有线程数。

运行 C++ 程序示例：

```
younghojan@younghojan-XPS-15-7590:/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/tutorial-code$ ../build/bin/clang++ -o run_time_system_functions run_time_system_functions.cpp -fopencilk
run_time_system_functions.cpp:10:5: error: use of undeclared identifier '__cilkrts_set_param'
    __cilkrts_set_param("nworkers","20");
    ^
run_time_system_functions.cpp:15:21: warning: '__cilkrts_get_worker_number' is deprecated [-Wdeprecated-declarations]
    int workerNum = __cilkrts_get_worker_number();
                    ^
/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/build/lib/clang/10.0.1/include/cilk/cilk_api.h:11:66: note: '__cilkrts_get_worker_number' has been explicitly marked deprecated here
extern unsigned __cilkrts_get_worker_number(void) __attribute__((deprecated));
                                                                 ^
run_time_system_functions.cpp:18:24: error: use of undeclared identifier '__cilkrts_get_total_workers'; did you mean '__cilkrts_get_tls_worker'?
    int totalWorkers = __cilkrts_get_total_workers();
                       ^~~~~~~~~~~~~~~~~~~~~~~~~~~~
                       __cilkrts_get_tls_worker
/media/younghojan/Study/undergraduate/junior/SEM1/软件系统优化/homework/hw5/build/lib/clang/10.0.1/include/cilk/cilk_api.h:12:26: note: '__cilkrts_get_tls_worker' declared here
struct __cilkrts_worker *__cilkrts_get_tls_worker(void);
                         ^
run_time_system_functions.cpp:18:9: error: cannot initialize a variable of type 'int' with an rvalue of type 'struct __cilkrts_worker *'
    int totalWorkers = __cilkrts_get_total_workers();
        ^            ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 warning and 3 errors generated.
```

又出错了，我麻了。

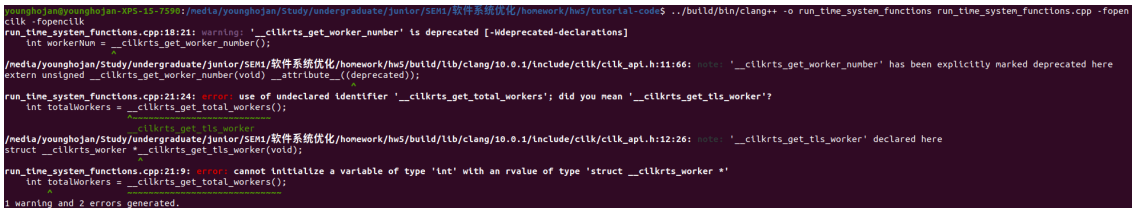
一个一个 error 来看：

- run_time_system_functions.cpp:10:5: error: use of undeclared identifier '__cilkrts_set_param'

意思是 __cilkrts_set_param 没有声明，但是 tutorial 中并没有要求声明这个函数。在 main 函数外声明该函数：

```
int __cilkrts_set_param(const char* name, const char* value);
```

尝试编译：



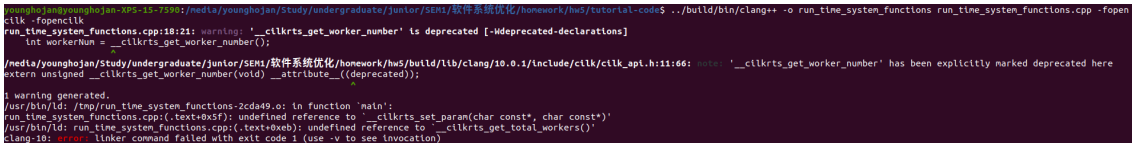
这个 error 消失了，姑且采用这个解决方案吧！

- run_time_system_functions.cpp:18:24: error: use of undeclared identifier '__cilkrts_get_total_workers'; did you mean '__cilkrts_get_tls_worker'?

  又是函数未声明的错误，在 main 函数外声明即可：

```
int __cilkrts_get_total_workers (void);
```

编译结果：



新错误出现了！

直接把 cilk_api.h 打开看看好了：

```
#ifndef _CILK_API_H
#define _CILK_API_H
#ifdef __cplusplus
extern "C" {
#endif

extern int __cilkrts_is_initialized(void);
extern int __cilkrts_atinit(void (*callback)(void));
extern int __cilkrts_atexit(void (*callback)(void));
extern unsigned __cilkrts_get_nworkers(void);
extern unsigned __cilkrts_get_worker_number(void)
__attribute__((deprecated));
struct __cilkrts_worker *__cilkrts_get_tls_worker(void);


#if defined(__cilk_pedigrees__) || defined(ENABLE_CILKRTS_PEDIGREE)
#include <inttypes.h>
typedef struct __cilkrts_pedigree {
    uint64_t rank;
    struct __cilkrts_pedigree *parent;
} __cilkrts_pedigree;
extern __cilkrts_pedigree __cilkrts_get_pedigree(void);
extern void __cilkrts_bump_worker_rank(void);
extern uint64_t __cilkrts_get_dprand(void);
#endif // defined(__cilk_pedigrees__) || defined(ENABLE_CILKRTS_PEDIGREE)
```

```
#undef VISIBILITY

#ifdef __cplusplus
}
#endif

#endif /* _CILK_API_H */
```

`int __cilkrts_set_param(const char* name, const char* value);` 和 `int __cilkrts_get_total_workers (void);` 没有在 cilk_api.h 中定义。可能 tutorial 的代码已经过时了？我不理解！

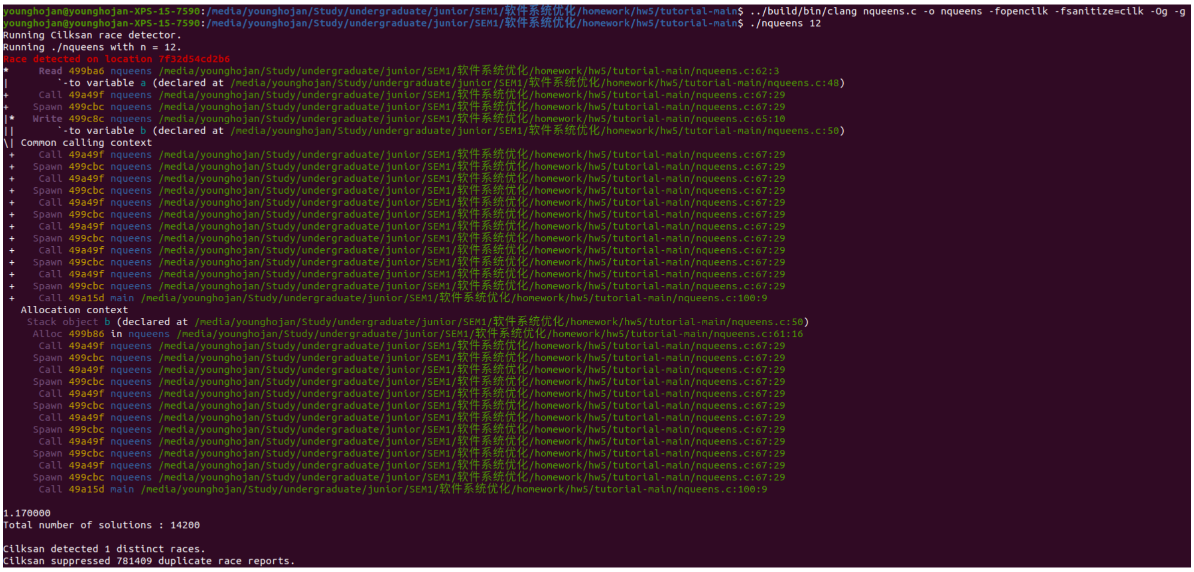
在 https://github.com/OpenCilk/infrastructure 找到了解释：

> OpenCilk 1.0 is largely compatible with Intel's latest release of Cilk Plus. Unsupported features include:
>
> - Cilk Plus array-slice notation.
> - Certain Cilk Plus API functions, such as `__cilkrts_set_param()`.

阅读教程“Research and Teaching with OpenCilk”，了解 OpenCilk 的研发历程，尝试教程P31~P36 的 6 个 Demo，记录执行过程和结果

To compile a program with OpenCilk, pass the `-fopencilk` flag to Clang:

Add the `-fsanitize=cilk` compiler flag to enable the Cilksan determinacy-race detector:



Add the `-fcilktool=cilkscale` compiler flag to measure work and span using Cilkscale:

Annotate the code(qsort.c) using the Cilkscale API:

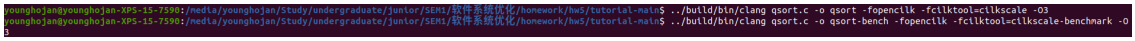
```
#include <cilk/cilkscale.h>

int main(int argc, char **argv) {
    …
    wsp_t start, end;
    start = wsp_getworkspan();
    sample_qsort(a, a + n);
    end = wsp_getworkspan();
    …
    wsp_dump(wsp_sub(end, start), "sample_qsort");
    …
}
```

Download OpenCilk productivity-tools repository:

```
git clone https://github.com/OpenCilk/productivity-tools.git
```

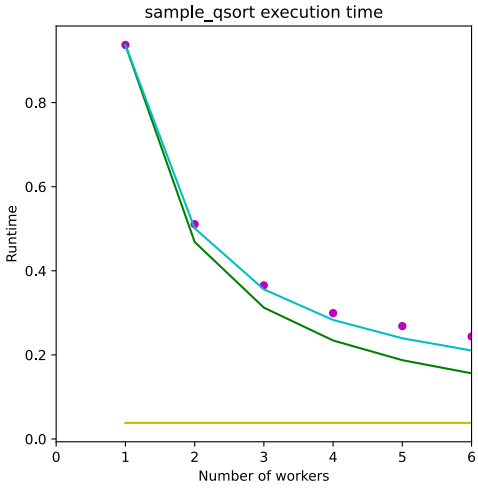
Compile the program twice,

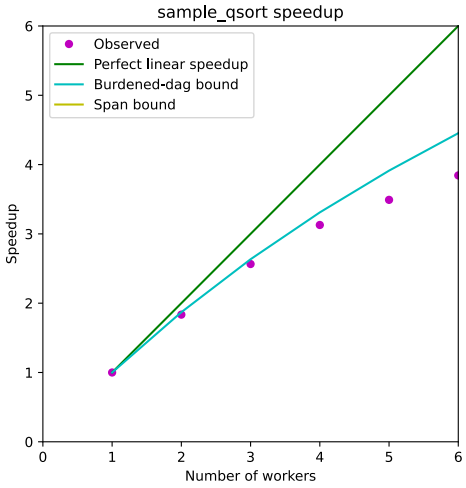
- once with –fcilktool=cilkscale, and
- once with –fcilktool=cilkscale-benchmark:

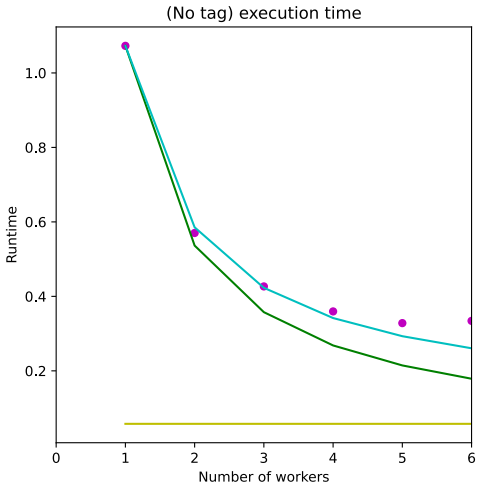


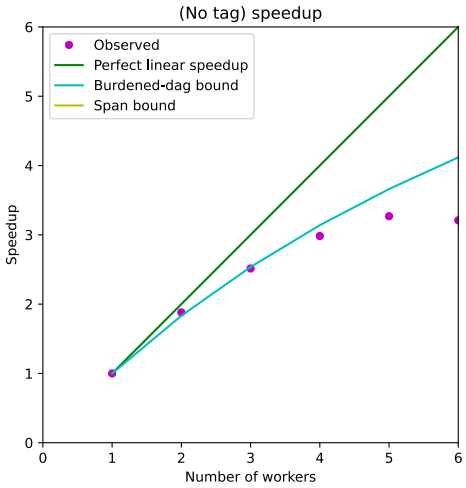
Run the program with the visualizer:

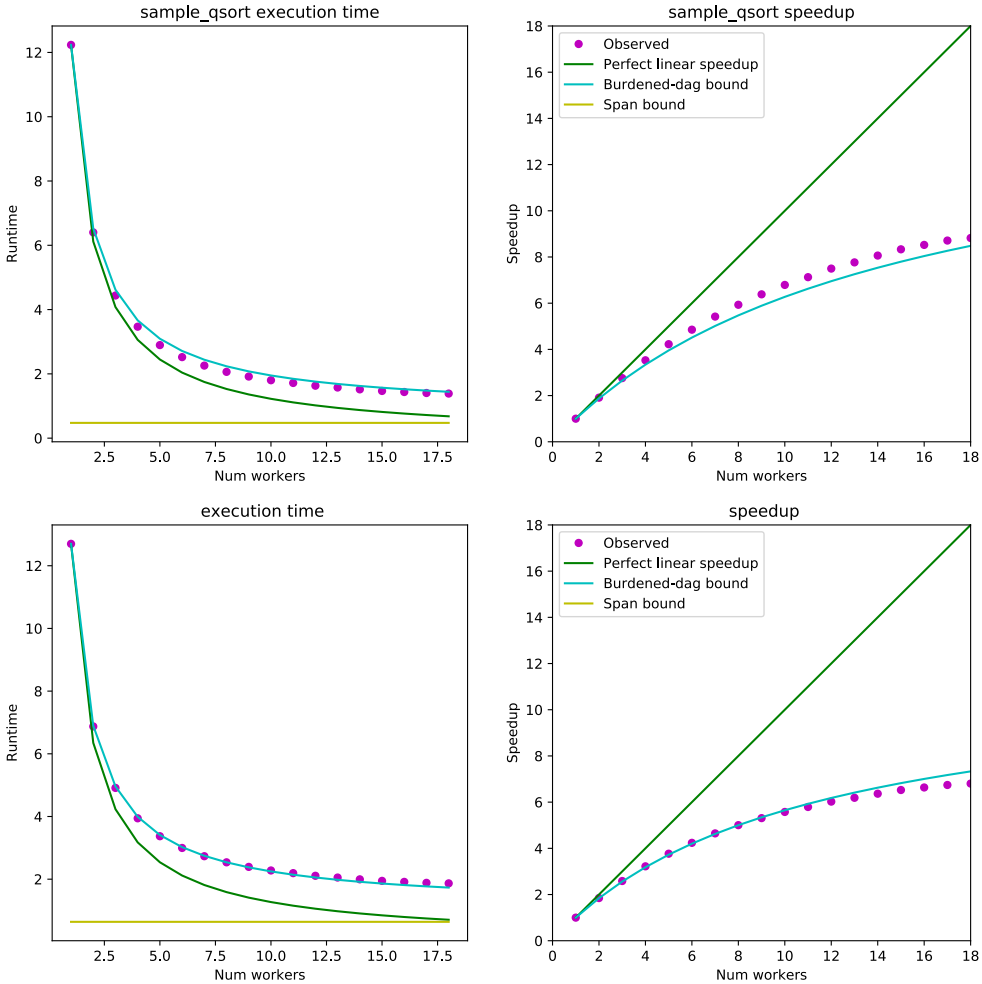
Open plot.pdf/out.pdf to view the performance plot.


sample_qsort execution time
Runtime
Number of workers
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
5
6


sample_qsort speedup
Observed
Perfect linear speedup
Burdened-dag bound
Span bound
Speedup
Number of workers
0
1
2
3
4
5
6


(No tag) execution time
Runtime
Number of workers
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6


(No tag) speedup
Observed
Perfect linear speedup
Burdened-dag bound
Span bound
Speedup
Number of workers
0
1
2
3
4
5
6


sample_qsort execution time
Runtime
Num workers
sample_qsort speedup
Observed
Perfect linear speedup
Burdened-dag bound
Span bound
Speedup
Num workers
execution time
Runtime
Num workers
speedup
Observed
Perfect linear speedup
Burdened-dag bound
Span bound
Speedup
Num workers