TaskScheduler

base/task_scheduler/post_task.h

Available everywhere.

BEFORE

components/something/something.cc

```
void DoSomething(scoped_refptr<base::TaskRunner> task_runner) {
  task_runner->PostTask(...);
}
```

chrome/chrome.cc

```
DoSomething(content::BrowserThread::GetBlockingPool());
```

AFTER

components/something/something.cc

```
void DoSomething() {
  base::PostTask(...);
}
```

chrome/chrome.cc

```
DoSomething();
```

Saved 100 lines of code in one component:
https://codereview.chromium.org/2885173002/

base/task_scheduler/post_task.h

Handles priorities.

BEFORE

```
// Use a 5 seconds delay because we'll probably not be
// doing anything important in 5 seconds.
task_runner->PostDelayedTask(
  FROM_HERE,
  base::BindOnce(...),
  base::TimeDelta::FromSeconds(5));
```

True story: chrome/browser/intranet_redirect_detector.cc

AFTER

```
// Will be scheduled when the browser is not doing anything
// important.
base::PostTaskWithTraits(
  FROM_HERE,
  {base::TaskPriority::BACKGROUND},
  base::BindOnce(...));
```

Solves shutdown hangs.

BEFORE

```
// I hope this won't delay shutdown!
file_thread->PostTask(...);
```

AFTER

```
// Shutdown will not block on this task.
base::PostTaskWithTraits(
  FROM_HERE,
  {base::TaskShutdownBehavior::CONTINUE_ON_SHUTDOWN},
  base::BindOnce(...));
```

Status of the migration

API	Removed from
base::WorkerPool	54 / 55 files
base::SequencedWorkerPool	223 / 391 files
BrowserThread::FILE	16 / 335 files Fixit week!
BrowserThread::DB	0 / 45 files
BrowserThread::FILE_USER_BLOCKING	1 / 7 files
BrowserThread::CACHE	0 / 10 files
BrowserThread::PROCESS_LAUNCHER	0 / 13 files

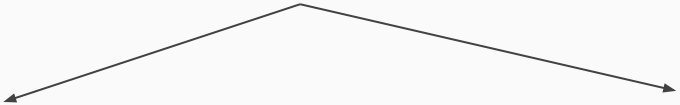
Migrating code to Task Scheduler

Migrate a BrowserThread::PostTask call site

1. Find a BrowserThread::PostTask call site to migrate.

```
BrowserThread::PostTask(
    BrowserThread::FILE, FROM_HERE,
    base::BindOnce(&ReadLastShutdownFile, type, num_procs, num_procs_slow));
```

2. Do other BrowserThread::PostTask call sites need to be sequenced with the BrowserThread::PostTask call site to migrate?
   - Look for other instances of BrowserThread::PostTask in the same file or same directory.



Yes
Go to “Migrate a **sequenced** BrowserThread::PostTask” call site

No
Go to “Migrate a **parallel** BrowserThread::PostTask” call site

Migrate a **parallel** BrowserThread::PostTask call site

3. Change BrowserThread::PostTask -> base::PostTaskWithTraits.

```
- BrowserThread::PostTask(
-      BrowserThread::FILE, FROM_HERE,
       base::BindOnce(&ReadLastShutdownFile, type, num_procs, num_procs_slow));
```

You'll need to figure out which traits are appropriate.

```
+ base::PostTaskWithTraits(
+     FROM_HERE, {base::MayBlock(), base::TaskPriority::BACKGROUND},
      base::BindOnce(&ReadLastShutdownFile, type, num_procs, num_procs_slow));
```

Migrate a **sequenced** BrowserThread::PostTask call site

3. Create a SequencedTaskRunner and store it somewhere that can be accessed from all call sites that need to be sequenced.

```
class MyClass {
 public:
  ...

 private:
  const scoped_refptr<base::SequencedTaskRunner> task_runner_ =
      base::CreateSequencedTaskRunnerWithTraits(
          {base::MayBlock(), base::TaskPriority::BACKGROUND});
};
```

If the PostTask call sites don't have access to a common object, you may need Lazy*TaskRunner (base/task_scheduler/lazy_task_runner.h)

Migrate a **sequenced** BrowserThread::PostTask call site

4. Migrate all BrowserThread::PostTask call sites that need to be sequenced to PostTask() calls on your new SequencedTaskRunner.

```
void MyClass::Foo() {
  BrowserThread::PostTask(
      BrowserThread::FILE, FROM_HERE,
      callback);
}

void MyClass::Bar() {
  BrowserThread::PostTask(
      BrowserThread::FILE, FROM_HERE,
      callback);
}
```

```
void MyClass::Foo() {
  task_runner_->PostTask(
     FROM_HERE, callback);

}

void MyClass::Bar() {
  task_runner_->PostTask(
     FROM_HERE, callback);

}
```

Migrate a BrowserThread::PostTask call site

5 . Remove DCHECK_CURRENTY_ON checks in migrated tasks.

```
void MyClass::Callback() {
  DCHECK_CURRENTLY_ON(BrowserThread::FILE);
  base::ThreadRestrictions::AssertIOAllowed();
  ...
}
```

The task does not run on the FILE thread anymore.

If the task does synchronous IO operations, check that this is allowed on the current thread. This check will succeed in tasks posted to TaskScheduler with the base::MayBlock() trait.

Migrate a BrowserThread::PostTask call site

6. Upload a CL.
7. Verify that it passes the CQ dry run.
8. Get a review.
9. Land the CL.