Rust Zürichsee – August 2017

Location sponsor: Liip.ch

Organized by: Stefan Schindler @dns2utf8

Slides: https://github.com/rust-zurichsee/meetups

Agenda

Upcomming Events

- **RustFest.eu → 30.09. - 01.10. @ETHz**
- **September Meetup → 04.09. 19:00 @Coredump**

Main topics tonight

- **Admin**
- **ThreadPool**
- **Cargo helpers**

Admin

Verein Rust Zürichsee

- founded on the 10. july 2017

- looking for

*** location sponsors**

*** pizza bill sponsors**

Threadpool

- Worker threads eagerly created
- Job queue uses mutex
- Pool is joinable 😀
- All the docs → https://docs.rs/threadpool/

Synchronisation – Channel

```
let (tx, rx) = channel();
```

Pro:

- **Works like a pipe**
  - Work on already completed jobs
  - Caution: flooding a channel with small messages
- **Group tasks**

Con:

- **Must drop initial sender to complete iterator**
  - Work-around: `rx.iter().take( N ). …`

Synchronisation – Channel Example

```
use threadpool::ThreadPool; use std::sync::mpsc::channel;

let n_workers = 4; let n_jobs = 8;
let pool = ThreadPool::new(n_workers);

let (tx, rx) = channel();
for _ in 0..n_jobs {
    let tx = tx.clone();
    pool.execute(move|| {
        tx.send(1).unwrap();
    });
}

drop(tx);           assert_eq!(8, rx.iter().fold(0, |a, b| a + b));
```

Synchronisation – Barrier

```
let barrier = Arc::new(Barrier::new(n_jobs + 1));
barrier.clone() → move || { }
```

Pro:

- **Group tasks**

Con:

- **Potentially deadlock**
  - `assert!(n_jobs ⩽ n_workers)`

Synchronisation – Barrier Example

```rust
use threadpool::ThreadPool;
use std::sync::{Arc, Barrier};
use std::sync::atomic::{AtomicUsize, Ordering};

// create at least as many workers as jobs or you
// will deadlock yourself
let n_workers = 42;
let n_jobs = 23;
let pool = ThreadPool::new(n_workers);
let an_atomic = Arc::new(AtomicUsize::new(0));

assert!(n_jobs <= n_workers,
        "too many jobs, will deadlock");

// a barrier to wait for all jobs plus the initiator
let barrier = Arc::new(Barrier::new(n_jobs + 1));
```

```rust
for _ in 0..n_jobs {
    let barrier = barrier.clone();
    let an_atomic = an_atomic.clone();

    pool.execute(move|| {
        // do the heavy work
        an_atomic.fetch_add(1, Ordering::Relaxed);

        // then wait for the other threads
        barrier.wait();
    });
}

// wait for the threads to finish the work
barrier.wait();
assert_eq!(23, an_atomic.load(Ordering::SeqCst));
```

Synchronisation – Join

```
pool.join();
```

Pro:

- **Can submit new jobs while joining**
- **Many threads can wait for a pool**

Con:

- **Never join from within the pool → Deadlock**

Synchronisation – Join Example 0

```
use threadpool::ThreadPool;

let pool = ThreadPool::new(2);
pool.execute(|| println!("hello"));
pool.execute(|| println!("world"));
pool.execute(|| println!("foo"));
pool.execute(|| println!("bar"));

pool.join();
```

Synchronisation – Join Example 1

```
use threadpool::ThreadPool; use std::sync::Arc; use std::sync::atomic::{AtomicUsize, Ordering};

let pool = ThreadPool::new(8);
let test_count = Arc::new(AtomicUsize::new(0));

for _ in 0..42 {
    let test_count = test_count.clone();
    pool.execute(move || {
        test_count.fetch_add(1, Ordering::Relaxed);
    });
}

pool.join();
assert_eq!(42, test_count.load(Ordering::Relaxed));
```

More examples

The Rust Cookbook:

https://rust-lang-nursery.github.io/rust-cookbook/

Next, the architecture of ThreadPool

Synchronisation – Join Code – ThreadPool

```
/// Abstraction of a thread pool for basic parallelism.
#[derive(Clone)]
pub struct ThreadPool {
    // How the threadpool communicates with subthreads.
    //
    // This is the only such Sender, so when it is dropped
    // all subthreads will quit.
    jobs: Sender<Thunk<'static>>,
    shared_data: Arc<ThreadPoolSharedData>,
}
```

Synchronisation – Join Code – Shared data

```
struct ThreadPoolSharedData {
    name: Option<String>,
    job_receiver: Mutex<Receiver<Thunk<'static>>>,
    empty_trigger: Mutex<()>,
    empty_condvar: Condvar,
    queued_count: AtomicUsize,
    active_count: AtomicUsize,
    max_thread_count: AtomicUsize,
    panic_count: AtomicUsize,
}
```

Synchronisation – Join Code – Notify

```
/// impl ThreadPoolSharedData: Notify all observers joining
/// this pool if there is no more work to do.
    fn no_work_notify_all(&self) {
        if !self.has_work() {
            *self.empty_trigger.lock().unwrap();
            self.empty_condvar.notify_all();
        }
    }
```

Synchronisation – Join Code – Observe

```
pub fn join(&self) {
    while self.shared_data.has_work() {
        let mut lock = self.shared_data
                           .empty_trigger.lock().unwrap();
        while self.shared_data.has_work() {
            lock = self.shared_data
                       .empty_condvar.wait(lock).unwrap();
        }
    }
}
```

Questions?

Cargo helpers

Rust Zürichsee

August 2017 Meetup

Cargo helpers

- **cargo-outdated**
  - List outdated crates inside a project. Works with `Cargo.lock` and `Cargo.toml`
- **cargo-tree**
  - Generates Ascii-Art
- **cargo-update**
  - Updates programs installed with `cargo install`
  - Command: `cargo install-update --all`
- **rustfmt**
  - Format all the source code
  - Includes cargo-fmt

Rust Zürichsee – August 2017

Location sponsor: Liip.ch

Organized by: Stefan Schindler @dns2utf8

Slides: https://github.com/rust-zurichsee/meetups