Thread pools and iterators

Stefan Schindler (@dns2utf8)

January 15, 2018

Rust Zürichsee, Schweiz CH

Inhalt

1. About
2. Modes of execution
3. Implementation
4. Examples
5. Code to iterators
6. Some pitfalls
7. Questions (max 10min)
8. Workshop time

About

Timetable

- now => Talk
- 20:00 => Questions
- 20:10 => Happy hacking
- 21:00 => Closing
- tomorrow => ???
- the day after => Parallelize the World!

About:me

Hi my name is Stefan and I do Computer Science.

I organze

- RustFest.eu Paris: Tentatively May 26th & 27th (but don't book yet!) with impldays (around the weekend)
- Meetups in and around Zürich
- Illuminox.ch (Swiss alps in July 2018)

Some of my side projects

- rust threadpool
- Son of Grid Engine (SGE) interface
- run your own infrastructure - DNS, VPN, Web, ...

What we will learn tonight

- The different modes of execution
- Single vs. Multi Threading
- How to synchronize pools
- How to translate linear code into parallel code

Modes of execution

Programming is ...

... about solving problems

Examples:

- Copy data
- Enhance audio
- Distribute messages
- Store data
- Prepare thumbnails

Key is understanding the problem

Single thread

How to do more than one thing at the time?

- Linear if tasks are short enough
- Polling
- Event driven (select/epoll)
- Hardware SIMD

Multi Threading

Let's add another level of abstraction

- spawn / join: handle lists of JoinHandles
- pools
  - job queue (the one we look at)
  - Workstealing (rayon)
  - futures

New problems: synchronization and communication

Implementation

Send and Sync

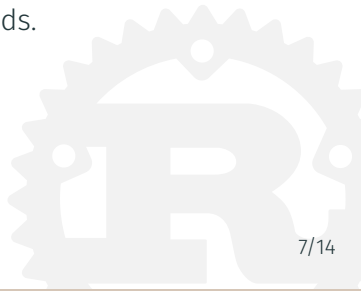
Rusts ”pick three” (safety, speed, concurrency)

`Trait std::marker::Send`

Types that can be transferred across thread boundaries.

`Trait std::marker::Sync`

Types for which it is safe to share references between threads.



Crates

Let's add another level of abstraction

- spawn / join: handle lists of JoinHandles
- pools
  - job queue (the one we look at)
  - Workstealing (rayon)
  - futures

New problems: synchronization and communication

Examples

Channel

```
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).expect("channel will be there");
    });
}
drop(tx);

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

Code to iterators

Collect from channnel

v_len stores how many elements

```
for _ in 0..v_len {
  if let Some(pi) = rx.recv().unwrap() {
    g.pictures.push( pi );
  } else {
    // Abort because of some error in the thread
    return;
  }
}
```

Collect from channnel

```
for pi in rx.iter() {
  if let Some(pi) = pi {
    g.pictures.push( pi );
  } else {
    // Abort because of some error in the thread
    return;
  }
}
```

Collect from channnel

```
rx.iter().for_each(|pi| {
  if let Some(pi) = pi {
    g.pictures.push( pi );
  } else {
    // Abort because of some error in the thread
    return;
  }
});
```

Collect from channnel

```
g.pictures = rx.iter().map(|pi| {
  if let Some(pi) = pi {
    Ok( pi )
  } else {
    // Abort because of some error in the thread
    Err( () )
  }
}).collect::<Result<Vec<PictureInfo>, ()>>().unwrap();
```

Some pitfalls

TcpStream with SGE array jobs

Question: How many connections will each client open

```
peer_streams = map.values()
    .filter(|s| s.is_some())
    .map(|s| s.unwrap())
    .map(|(addr, data_port)|
        TcpStream::connect(
            SocketAddr::new(addr, data_port)))
    .filter(|s| s.is_ok())
    .map(|s| s.unwrap())
    .collect();
```

Questions (max 10min)

Workshop time

Thank you for your attention!

Stefan Schindler @dns2utf8

Happy hacking! Please ask questions!

Slides: `https://github.com/dns2utf8/thread-pools-and-iterators`