L6: Future, Promises and Rust Intro

Rachata Ausavarungnirun

(rachata.a@tggs.kmutnb.ac.th)

October 16th, 2020

Architecture Research Group

Computer Engineering, TGGS

Concurrent+Asynchronous Programs

- What does it mean for a program to have concurrency?
- What does it mean for a program to be asynchronous?
- Why does functional programming model fit well?
    - Recall dataflow paradigm?

Example

- Let’s count the number of the occurrence of a list of web URLs
- See code example on canvas
- Let’s break this down and see what is the blocking calls
- Blocking call: a part of the program that has to finish before the next part begins
  - Why is this bad for the performance of parallel programs?

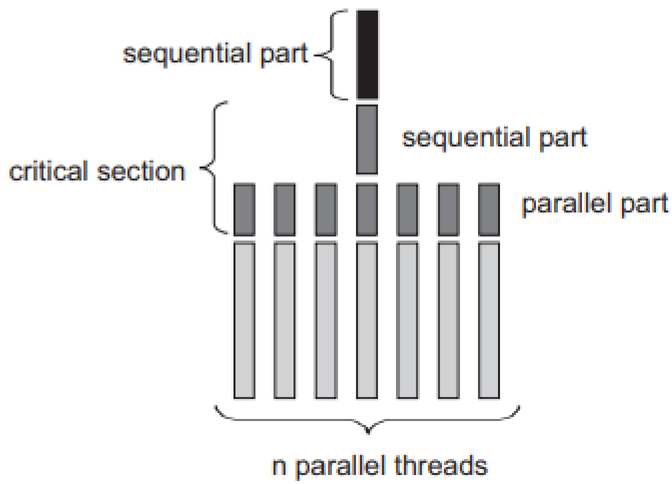
Amdahl’s Law

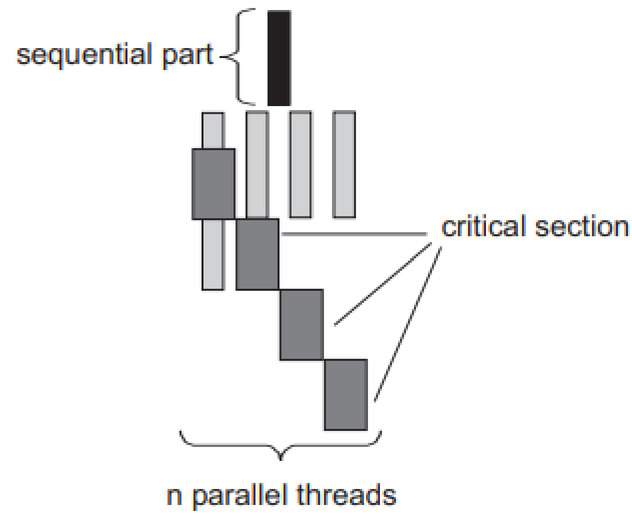
- Amdahl’s Law:
  - The performance speedup in parallel programs depend on **both the serial parts and the parallel parts**
    - **Serial portion determined by the algorithm**
  - This can limit the benefit of parallelizing your program
  - Serial portion of the code becomes the critical performance bottleneck in parallel programs
  - Gene Amdahl, “Validity of the single processor approach to achieving large scale computing capabilitie”, AFIPS 1967.

- Critical section → Parts of program execution that require synchronization/need to resolve contentions
  - Amdahl’s Law did not mention this part

Breaking Down Critical Sections

- Critical section can be broken down into two cases
  - With synchronization (What can cause this?)
  - Without synchronization





- Eyerman and Eeckhout, “Modeling Critical Sections in Amdahl’s Law and its Implications for Multicore Design”, ISCA 2010.

How to Handle Blocking

- Synchronously
  - Wait for the serial portion/blocking call to complete
  - Run the next processing task afterward
  - What are the benefits?
  - What are the downsides?

- Asynchronously
  - While waiting for the blocking process → Switch to an independent task
  - What are the benefits?
  - What are the downsides?

The Notion of Time

- Traditionally, functional programming does not focus on the notion of time
  - Why?
  - This is along the same logic as why “state” does not exist

- Scala supports this

- You can use Future[T] to encode the notion of time
  - This is a value that will eventually become available
  - This value can have multiple states depending on the time we request the value

The State of The Future

- In its simplest form, Future[T] can have two states
  - Completed/determined → Computation is complete and the value is available
  - Incomplete/undetermined → Computation is not complete

- Future[T] is a container/wrapper type
  - Represent a value “that will eventually” results in type T
  - If the computation go wrong (time out, die horribly, etc.), this Future[T] has an exception of sort
  - This is a write-once container
    - **Becomes immutable** once the computation is complete

Example using Fib

- import scala.concurrent.Future
- import scala.concurrent.ExecutionContext.Implicits.global

- OK-ish fib
  - def fib(n: Int): Long = if (n <= 2) 1 else fib(n-1) + fib(n-2)
- Better fib using Future
  - val f1 = Future { fib(45) }
- Now I can even run two of fib at the same time
  - val f2 = Future { fib(46) }
  - **They will be run completely in the background**

Accessing the Future

- Assume f1 and f2 are defined, you can do
- f1.onComplete { case **Success**(result) => println(result) }

- If not successful, you can do
- import scala.util.{Success, Failure}
  val f = Future { fib(49) }
  f.onComplete {
  	case Success(v) => println("good ${v}")
  	case Failure(e) => println("Error: " + e.getMessage)
  }

Waiting for the Future

- Say if you want Scala to wait for the Future[T] to becomes ready
- import scala.concurrent.duration._
  Await.ready(f, 1 minutes)

Use Cases

- I/O calls → wait for the user input (standard I/O)
- Long computation → Use Future[T] to represent the completion of this task while running other tasks
- Database queries → Another instance of long processes
- RPC → Network latency is long, use Future[T] to represent the result of the remote procedure calls
- Timeout → Force a return of no result or empty result on Future[T] to represent a timeout

Futures vs. Promises

- Scala also supports promises
- A promise is a single-assignment variable which the future refer to
  - You can get the future with a promise, but not vice-versa
- Think of this as how you handle real-life promise
  - If you promise something to someone, you must keep it
  - If someone promise something to you, they should honor it in the future

Realizing Futures vs. Promises

- Ok … all the things we discussed so far are good ideas

- But, hardware is plain and stupid
  - How do I actually implement this concept?

- Why you should care?
  - This **bridge the abstraction** between your program to the runtime and then to the hardware

Thread Pools and Event Loops

- What is a thread?
- A thread pool is a collection of idle threads that the runtime can assign work to
    - The actual thread pool implementation handles creating the workers, manage the workers, and schedule tasks
        - Active area of research as new type of computing systems emerge

- An event loop is an underlying system layers that are specifically designed to handle certain tasks
    - File system (for I/Os)
    - Database system (for queries)
    - Web services
    - All of which relies on its implementation, libraries, frameworks

Utilizing Future/Promises

- How can we unblock things using future/promises?
- Using Future to pipeline program execution
  - What is pipelining?
- Let’s go back to our example earlier

Group Exercise

1. Explain what the code is doing?

2. Explain what Future is used for?

Parallel Programming

Systems and Parallel Programming

- Why is this typically hard?

- Systems programming
  - Hard to secure
  - Hard to multithread

- Parallel programs
  - Hard to detect data races
  - Hard to debug

Rust: Type Safety

- A well-define program
  - No undefined behavior on all cases of execution

- A type safe language
  - Every program is well defined

- Is C type safe?
  - Then, why should we use C?

Performance Is King

- Performance differs across languages
  - Why?

- Computer executes an the assembly code

- Language somehow translate your program into assembly code

- And we have not even touch how hardware can be very different as well
  - See CPU vs. GPU

Let’s Compare Performance

- Assume a program that check if a string only consist of whitespace

Ruby Performance

- class :: String
  ```
  def blank?
      /\A[[:space:]]*\z/ == self
  end
  ```
  end

- You can run 964K of these per second

C Performance

- Let's Optimize this on a C code
  - I am going to borrow the code from https://github.com/SamSaffron/fast_blank

- You can run 10.5M iterations in 1 sec

- This is ~10 times faster

Rust Performance

- **extern** “C” **fn** fast_blank(buf: Buf) -> bool {
  buf.as_slice().chars().all(|c| c.is_whitespace())
  }

- buf.as_slice() gets the string slice
- .chars() gets the iterator over each characters
- Then the rest just check if there are whitespaces

- This is 11M iterations/sec

Parallel Program in Rust

- Say I want to load many images from my input paths to all the images
- **fn** load_images(paths: &[PathBuf]) -> Vec<Image> {
```
  paths.iter()
    .map(|path| {
      Image::load(path)
    })
  .collect() }
```
- paths.iter() iterates over each path
- Then you load each path's image
- Create and return a vector of images
- **This is sequentially done**

Parallel Program in Rust

- I want to parallelize this
- ```
  extern crate rayon
  fn load_images(paths: &[PathBuf]) -> Vec<Image> {
    paths.par_iter()
      .map(|path| {
        Image::load(path)
      })
    .collect() }
  ```
- rayon is a data-parallel library that convert sequential execution into parallel execution
  - https://docs.rs/rayon/1.3.0/rayon/
- paths.par_iter() iterates over each path in parallel
  - We will go into the detail on what’s parallelizable later
  ```

Detecting Data Races

- Rust compiler will tell you
- ```
  fn load_images(paths: &[PathBuf]) -> Vec<Image> {
   let mut jpegs = 0;
   paths.par_iter()
     .map(|path| {
       if path.ends_with("jpeg") { jpegs += 1; }
       Image::load(path)
     })
     .collect();
  }
  ```
- Note: let binds a value to a variable
- This will not compile, why?
  ```

Detecting Data Races

- Rust compiler will tell you
-
```
fn load_images(paths: &[PathBuf]) -> Vec<Image> {
  let mut jpegs = 0;
  paths.par_iter()
    .map(|path| {
      if path.ends_with("jpeg") { jpegs += 1; }
      Image::load(path)
    })
    .collect();
}
```
- Note: let binds a value to a variable
- This will not compile, why?
  - Need to lock this (or use AtomicU32)

Side Note on Atomic

- What is an atom?

- What is an atomic operation?

- Why is this useful?

Reading Rust Code

- fn defines a function

Input variable name (mutable) and types

- fn gcd(mut n: u64, mut m: u64) -> u64 {

```
  assert!(n != 0 && m != 0);      This is a macro, ! asserts only if false
  while m != 0 {
    if m < n {      Note no parenthesis for the condition check is ok
      let t = m; m = n; n = t;      Create a local variable. Type is inferred
    }
    m = m % n;
  }
  n      You can also type return n, but the last line is the return value similar to Scala
}
```

Generics

- fn min<T: Ord>(a: T, b: T) -> T {
  if a <= b { a } else { b }
  }

```
fn min<T: Ord>(a: T, b: T) -> T {
  if a <= b { a } else { b }
}
```

- Ord means that T, which is a generic type have total ordering

Enumeration and Sum Types

- ```
  enum Option<T> {
    None,
    Some(T)
  }
  ```
- ```
  fn safe_div(n: i32, d: i32) -> Option<i32> {
    if d == 0 { return None; }
    return Some(n / d); }
  ```
- ```
  match safe_div(num, denom) {
    None => println!("No quotient."),
    Some(v) => println!("quotient is {}", v)
  }
  ```
  ```

Catching Errors using Result<T,E>

- Result<T, E> is basically
  - enum Result<T, E> {
    Ok(T),
    Err(E),
    }

- Let's use this to print the content of the directory and return the number of entries, or std::io::Error

Catching Errors using Result<T,E>

- ```
  use std::path::Path;
  fn list_directory(dir: &Path) -> std::io::Result<usize> {
    let mut count = 0;
    let entries = try!(std::fs::read_dir(dir));
    for entry_or_error in entries {
      let entry = try!(entry_or_error);
      println!("{:?}", entry.path());
      count += 1;
    }
    return Ok(count);
  }
  ```
- ```
  fn main() {
    let path = Path::new("/tmp");
    let dir_count = list_directory(&path);
    match dir_count {
      Ok(count) => println!("Total: {}", count),
      Err(err) => println!("Error: {:?}", err) }
  }
  ```

Memory Safety

- Rust provides three promises
- No null-pointer dereferencing
  - No Null value, use Option<T> or Result<T, E>
- No dangling pointers
  - What is a dangling pointer?
  - No garbage collectors as well
    - Use an ownership system
- No buffer overruns
  - No pointer arithmetic
  - Array in rust is not translated to pointers
  - Boundary checking at compile time

In-Class Exercise 11

- Install Rust and try it out
  - Write a program that print hello world

Review Session

Topics We Have Covered

- Evaluation rules
  - The substitution model
  - Termination
  - Evaluation Strategy (CBV, CBN)

- Conditionals

Topics We Have Covered

- Repetition
  - Recursion
  - Tail recursion
  - What if I want to do recursion on multiple functions?

- Compound data
  - List
  - Tuples
  - Making your own compound data → Tree

Topics We Have Covered

- Class vs. Objects
- Options
- Product type and Records
- Pattern matching

Topics We Have Covered

- Sum types
- Polymorphic types
- Exceptions
- Interaction between functions

Topics We Have Covered

- Folding and mapping
- Closure
- Scope (Lexical vs. Dynamic scope)
- Continuations
- Currying

Project Ideas

You Can Obviously Do This

- Implement some complex data structure in scala/rust
  - Graph
  - RBTree
    - Or, generally a B-tree
- Then, implement a sample of algorithm
  - Graph → BFS, Graph coloring, etc.
  - RBTree → Dictionary with O(logN) lookup, insertion, deletion

Projects from Other Classes

- Are you taking, or had already taken
  - OS
  - DBMS
  - Parallel algorithm
  - Scalable system
- These topics map well with OPL
  - We want parallelism (and performance)
  - We want scalability

- Play with Scala
- Play with Rust and implement a parallel version
- Try out OpenMP
- Try out CUDA (GPGPU)

More on Rust

- How many people here are annoyed by concurrency issues in parallel programming?
- How many people here are annoyed by the memory and dealing with memory?
- How many people here are annoyed by garbage collection?
  - If you use Go/JAVA, you are going to at some point …
- Rust is a programming language that is designed for
  - Safe concurrency
  - Memory safety
  - Still high performance
  - Syntactically similar to C++

Write Parallel Programs on GPUs

- For those who like challenges, try using the GPUs
  - Learning CUDA will definitely be useful on your resume
  - Please definitely check with me
    - You also need to find a GPU you can run the code on

Playing with Languages and Compilers

- You can analyze code and change the compiler
  - https://llvm.org/
  - https://scala-native.readthedocs.io/en/v0.3.9-docs/
    - This is Scala on top of the LLVM infrastructure

- This will also be very useful on your resume
  - You are trying to modify the compiler itself

- If you want to do this, please talk to me
  - There are some basic list of things you can try out
  - You need to set the milestones properly

Before We Leave Today

Next Weeks Plan

- Extra review/Q&A slot
  - October 20th Noon – 1 PM
  - October 22nd Noon – 1 PM
  - Recording will be posted on our Youtube channel
    - So you can watch these if you have conflicts

Exam Format

- Exam: take-home
  - All material up until now
- But, October 23rd is a holiday and a long weekend

- Instead of me giving you 24 hours, I will
  - Send out the questions on October 22nd
  - You have until October 26th midnight to finish
  - Same length as if you are working on a 24-hour exam
  - Open everything
  - I will be online in Webex at some point
    - The time will be announced next week (I need to check my schedule)
    - This will be live Q&A, just don't ask me what's the answer :p
  - Then, additional questions is through email/discord/msging