THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Distributed Systems CS6421

Intro to Distributed Systems and the Cloud Computing

Prof. Roozbeh Haghnazar

Slides Credit:

Prof. Tim Wood and Prof. Roozbeh Haghnazar

Prof. Roozbeh Haghnazar

- Started Programming in1991 with Commodore 64
- Played several roles in technology, such as Developer, Modeler, Designer, Architect, Leader, CTO, etc.
- Teach Software Eng., Distributed Systems, Data Base Design Principles, Data Visualization, Operating System.
- Data Science Lead in NWITS-USGS
- Tech Lead in Spirent Communications

About This Course

- Be prepared! (course prerequisites)
  - CSCI 6212 Algorithms (or undergrad algorithms course)
  - An undergraduate operating systems course
- Be involved!
  - “Raise hand”, ask questions , discuss, etc.
  - Asynchronous opportunities will be available
- Be ready to code!
  - You will need to use **Go, Python** for your assignments
  - Mostly group projects

Classes

- 2.5 hours is a long time for lectures!
  - We will try to break it up – discussions, demos, live coding
  - Some lectures may end early, with additional asynchronous material
- We want to make the best course we can for you!

Resources

- Slack: (linked from website, join after class)
- GitHub for collecting assignments
- Blackboard for grades, class meetings, and office hours

- Visual Studio Code – recommended IDE
  - Live share plugin allows group collaboration / help in office hours
- Repl.it – simple online editor for quick programming exercises
  - You can login with GitHub credentials if you want to save copies

Semester Outline

- Building Blocks
  - Introduction to Distributed System and Cloud
  - Scalable Execution: Processes, threads, VMs, containers, parallelism vs concurrency
  - Communication: RPC, Message Oriented, Stream Oriented
- **Principles** of Distributed Systems
  - Coordination: Synchronization, Consistency, and Consensus
  - Reliability: Replication and Fault Tolerance
  - Performance: Metrics and Modeling Large Scale Systems
- Distributed Systems in **Practice**
  - Grid Computing
  - Cloud Computing
  - Web, Mobile, and IoT

4 programming assignments
Midterm ???
Large group project

Introduction

- Computer systems are undergoing revolution.
- Two advances in technology changed the game
  - 8bit -> 16bit -> 32bit -> 64bit microprocessors
    - From a machine that cost $10M and executed 1 inst./sec we have come to machine that cost $1000 and execute 1 billion inst./sec
  - Computer networks LAN/WAN
    - From 64 Kbit/sec to Gigabit/Sec

INTRODUCTION

- If the automotive industry had advanced at the same rapid pace as computer science, today we could purchase a Rolls-Royce for just one dollar and get a billion miles per gallon

What is the Cloud?

What is the Cloud

- Giant warehouses
- 10s of thousands of servers
- Petabytes of storage
- 10s of thousands of Processor cores
- ….Interconnected….

Why Infrastructure?

- Why do we need this amount of infrastructures?
  - Encyclopedia Britannica
    - - 40,000+ articles
    - - 32 hard bound volumes (32,640 pages)
  - Wikipedia
    - - 5,512,202 articles (in English)
    - - More than **5 TB** of text (about 7,500 CDs)
    - -More than 2000 volumes

And then BIG Data

- Why do we need this amount of infrastructures?
  - Airbus A350
    - Contains around 6000 sensors across the entire plane that generates 2.5TB Data per day
  - Airbus A380-100
    - Expected to take the skies in 2020
    - Contains 10000 sensors just in each wings
  - Facebook
    - 20 TB photos each week
  - Google
    - 20000TB Data processing per day in 2008

And then BIG Data

- Google Search Statistics

The average figure of how many people use Google a day, which translates into at least 2 trillion searches per year, 3.8 million searches per minute, 228 million searches per hour, and 5.6 billion searches per day.

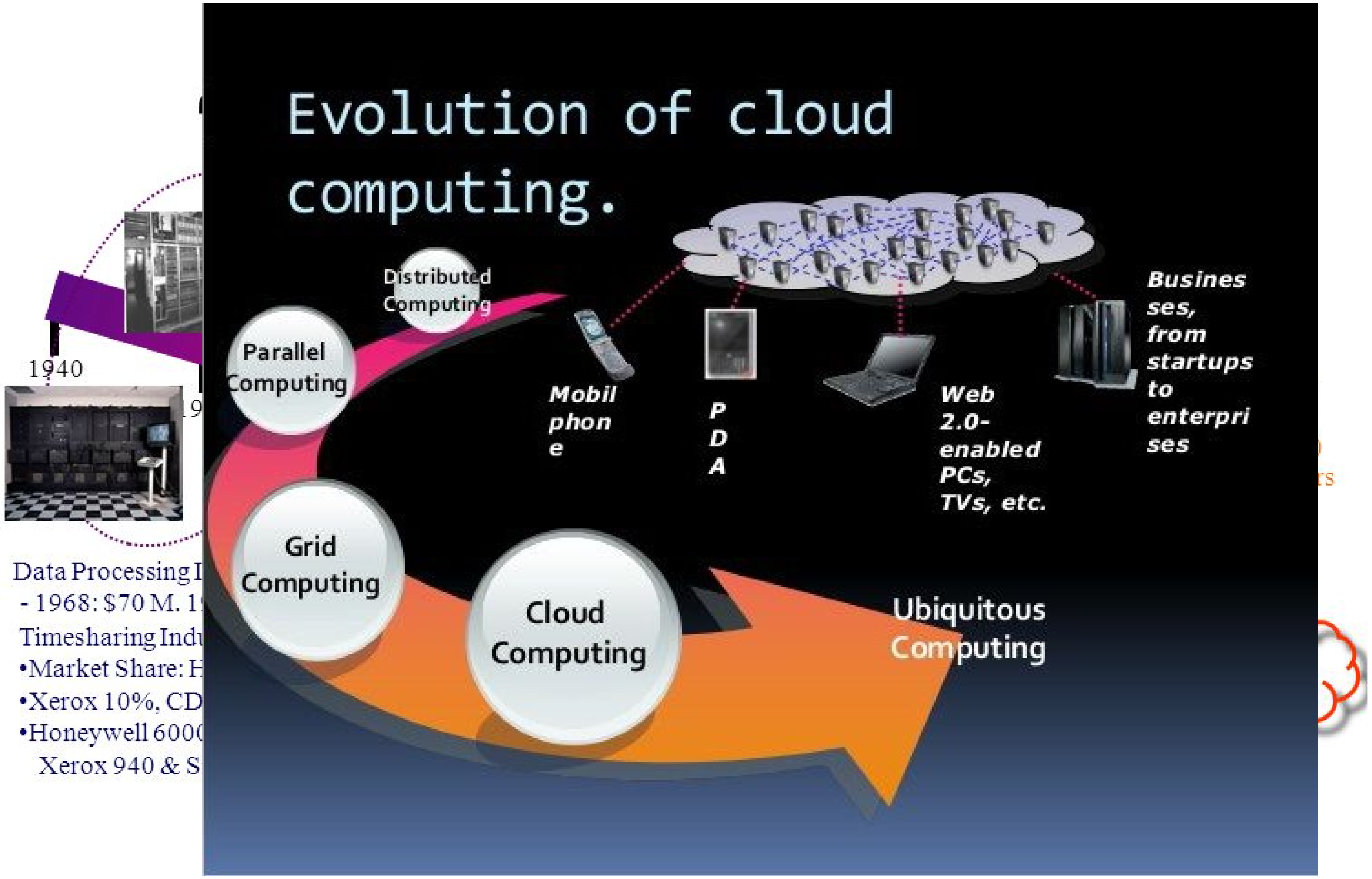
- How much data do we generate?

According to the Forbes statistics:

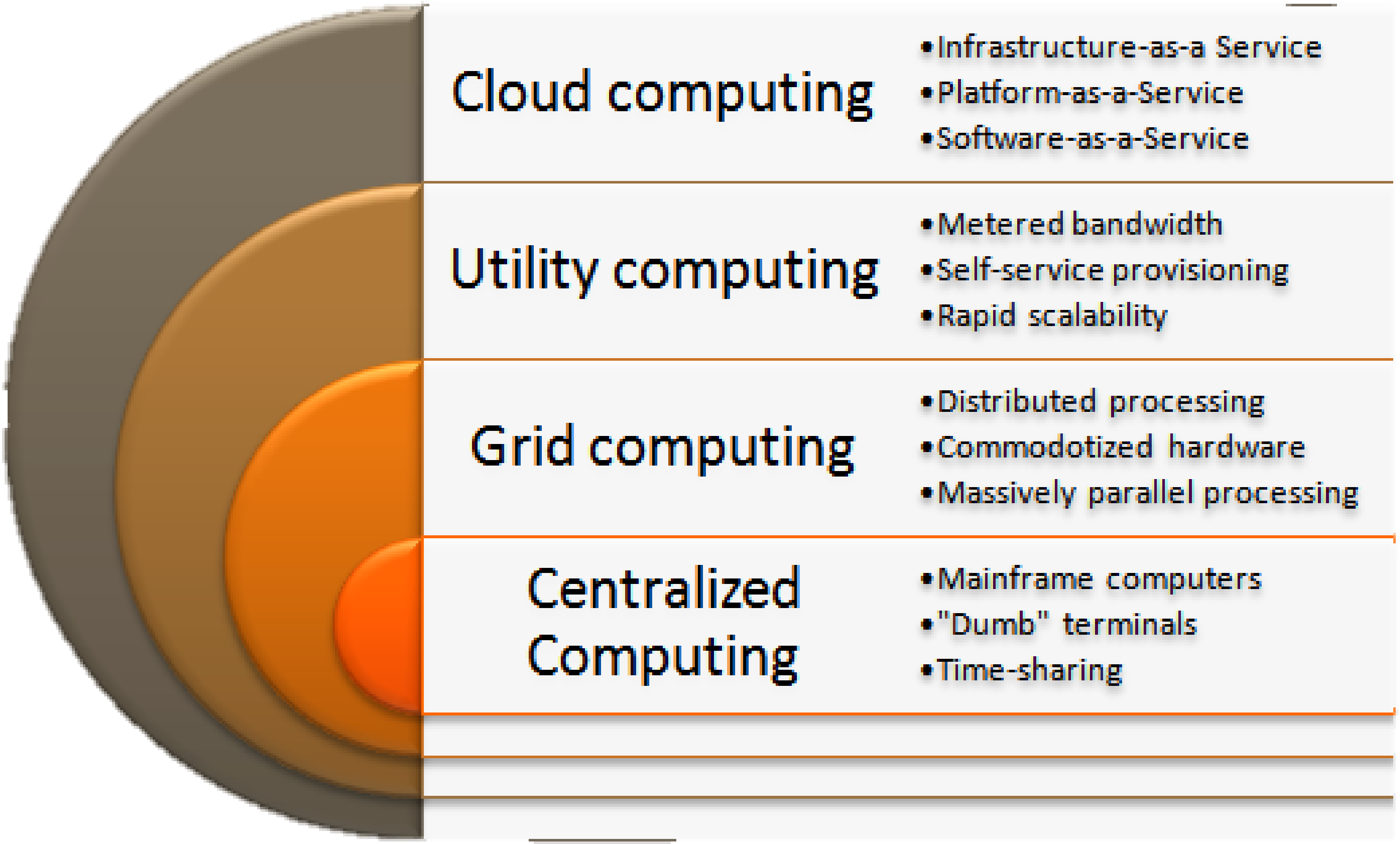
  - 2.5 quintillion bytes of data created each day
  - Over the last two years alone 90 percent of the data in the world was generated.

KB	Kilo Byte	1 thousand bytes
MB	Mega Byte	1 million bytes
GB	Giga Byte	1 billion bytes
TB	Tera Byte	1 trillion bytes
PB	Peta Byte	1 quadrillion bytes
EB	Exa Byte	1 quintillion bytes

History of Cloud Computing



History of Cloud Computing



What's New

- There are four new features in the new generation of distributed and cloud systems:
  - Massive Scale
  - On-Demand Access: Pay-as-you-go
  - Data Intensive Nature: MBs became PBs and XBs
  - New Cloud Programming Paradigms: Map/Reduce Hadoop, Unstructured Data

*aaS Classification

- HaaS : Hardware as a Service

  Hardware and backbone
- IaaS: Infrastructure as a Service

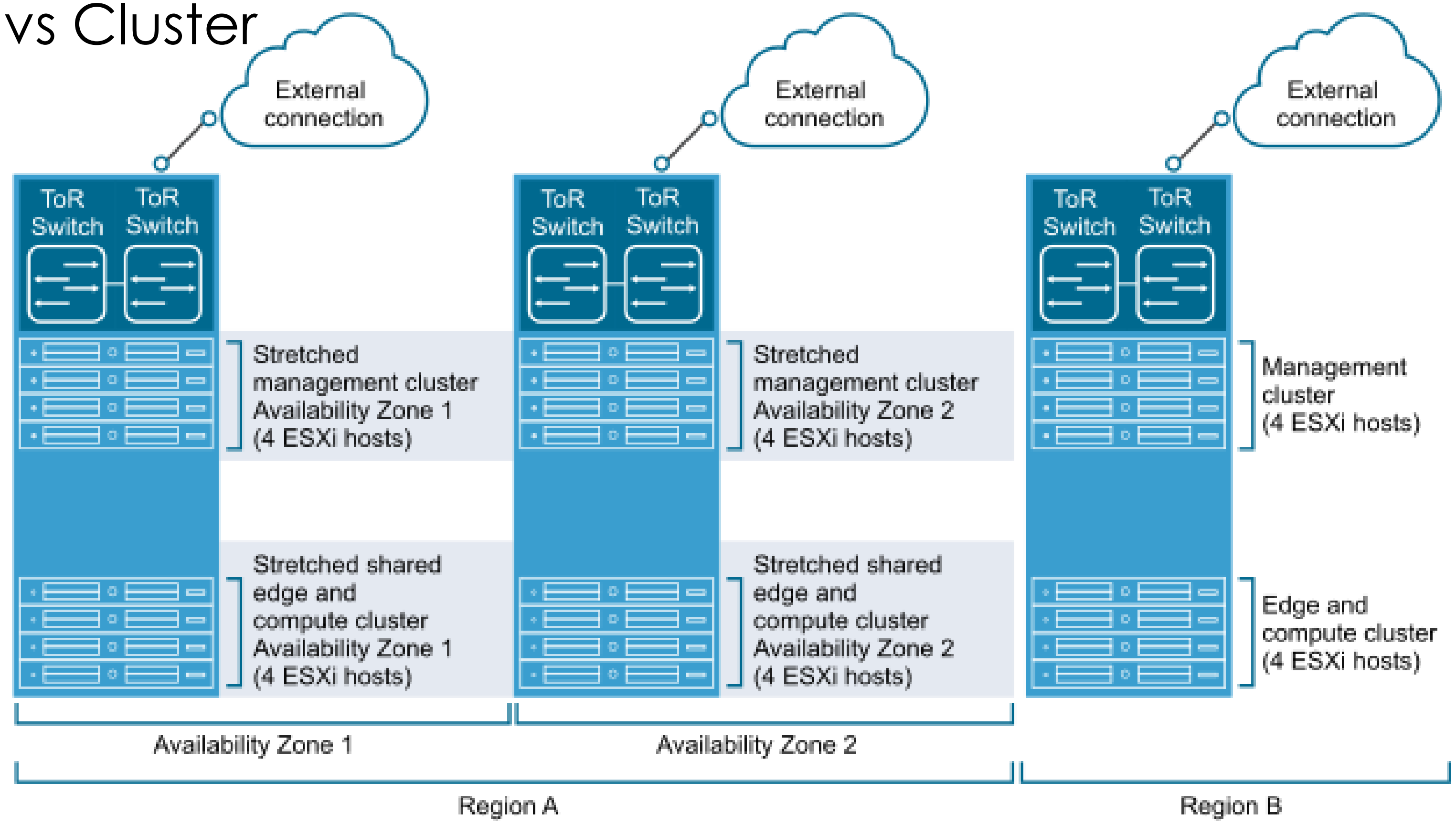
  AWS, Azure, GCP
- Paas: Platform as a Service

  Google App engine, AWS Elastic Beanstalk
- SaaS: Software as a Service

  Google Doc, Dropbox

Cloud is a …

- Cloud vs Distributed System vs Cluster
- Client Server Architecture



Cloud is a …

- Can we say " Cloud is a fancy word for a Distributed System?"

What is a Distributed System

- A distributed system is a collection of independent computers that appears to its users as a single coherent system. [Andrew Tanenbaum]
  - distributed system consists of components that are autonomous
  - users (be they people or programs) think they are dealing with a single system. (**Transparency)**
  - distributed systems should also be relatively easy to expand or scale.
  - Heterogeneity
  - Concurrency

Goals of DS

- Making resources accessible
- Distribution Transparency
  - Access
  - Location
  - Migration
  - Relocation
  - Replication
  - Concurrency
  - Failure
- Openness
- Scalability

Accessibility

- The main goal of a distributed system is to make it easy for the users and applications to access remote resources and to share them in a controlled and efficient way

Transparency

- **Transparency** in simple words is defined as the concealment from the user and the application programmer of the separation of components in a **distributed system**, so that the **system** is perceived as a whole rather than as a collection of independent components.

Openness

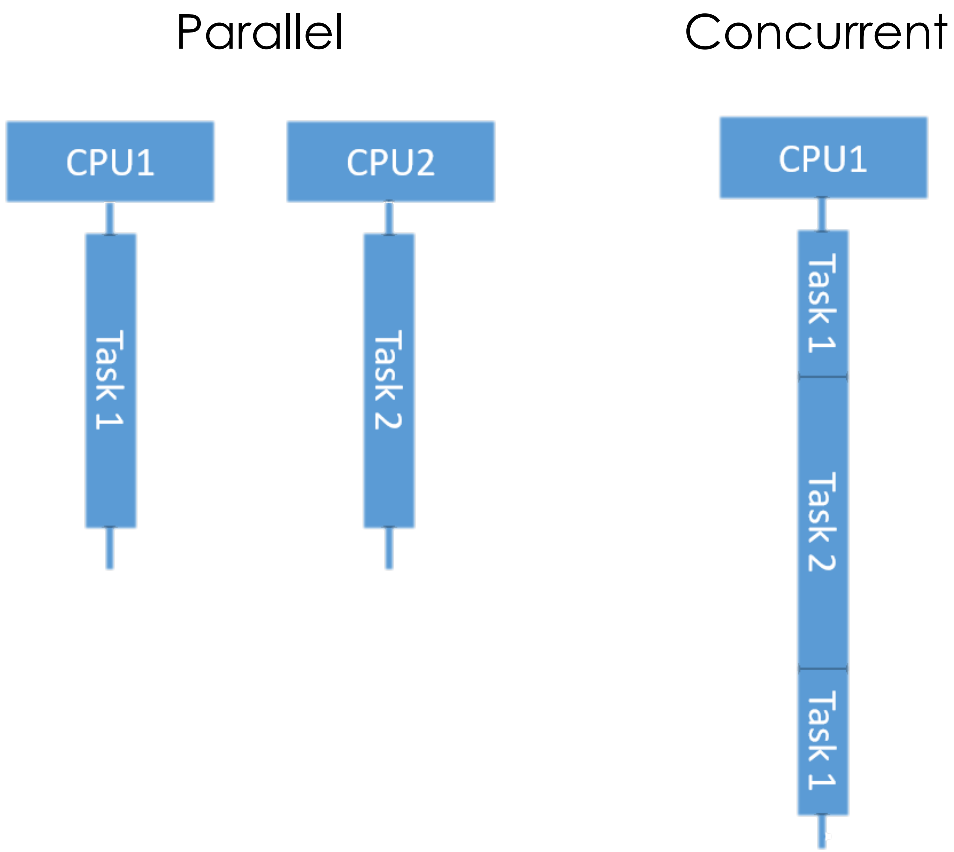
- An open distributed system is a system that offers services according to standard rules that describe the syntax and semantics of those services.

Scalability

- **Scalability** means you can increase or reduce the capacity, diversity, power or abilities of your system. It can be measured along at least three different dimensions:
  - A system can be scalable with respect to its size (add more users/resources to the system – can be consider as Scale up)
  - A geographically scalable system is one in which the users may lie far apart (Scale out)
  - A system can be administratively scalable. It means that it can still be easy to manage even if it spans many independent administrative organizations.

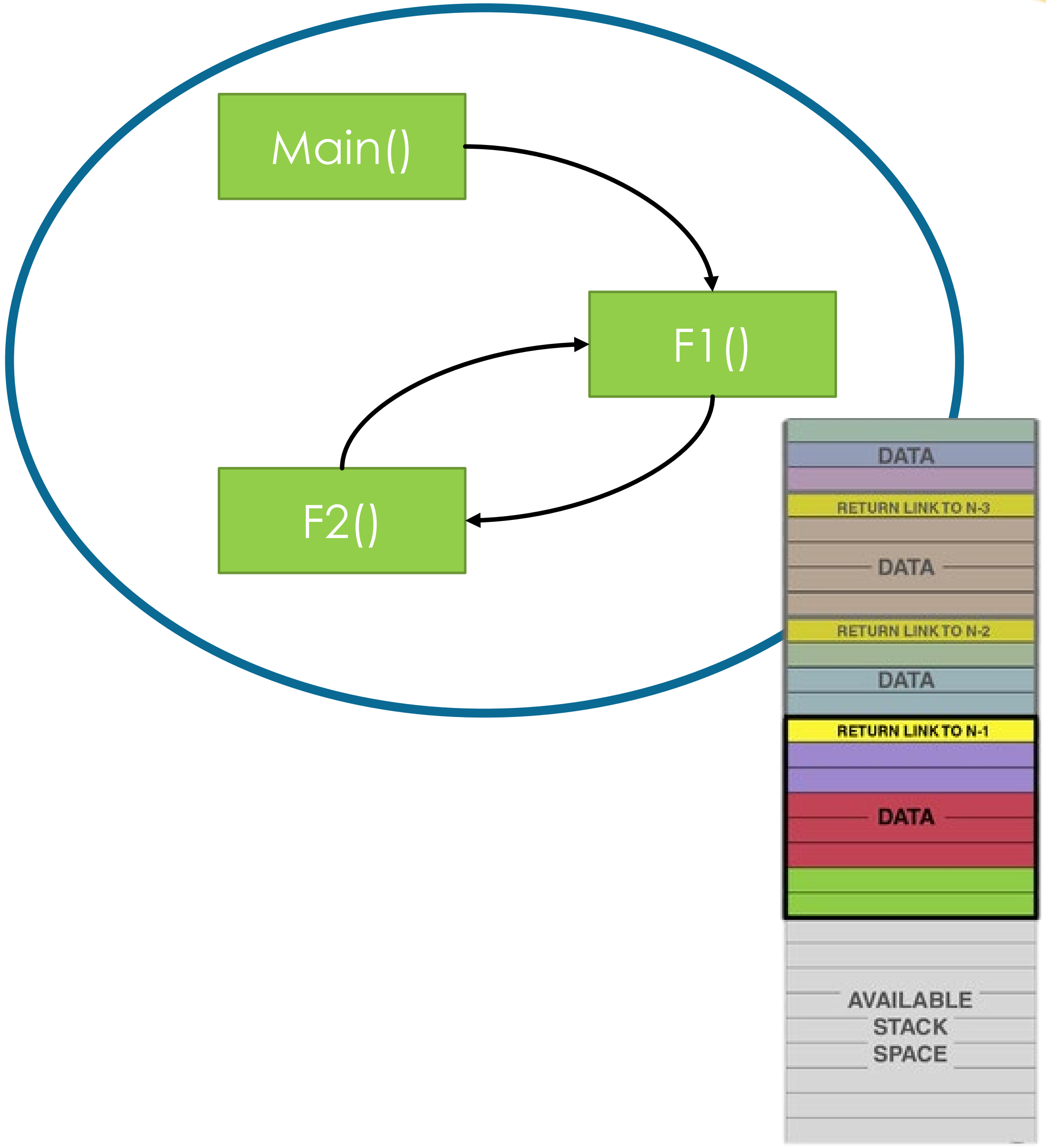
Concurrency vs Parallelism

- Concurrency considers the checkpoints
- Parallelism considers time of progresses



PROCESS

- Process
- Stack
- Program Counter
- Heap
- Etc.

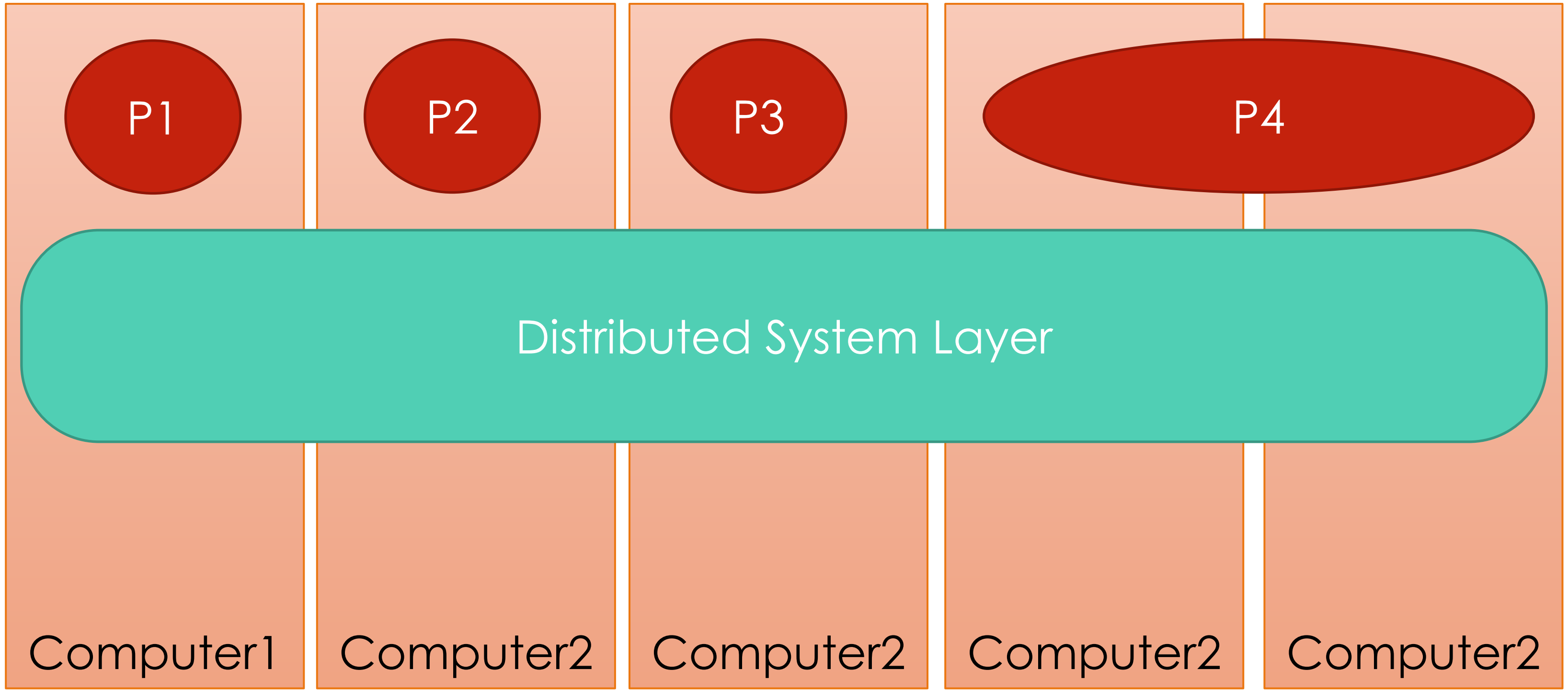


DISTRIBUTED ....

- Distributed System = Many Processes ?????

How can we Handle?

- Faster Computer Or Add Another Computer?



HW 1: Go Parallel Sum

Parallel Sum

- Assignment Goals:
  - Learn the basics of the Go programming language
  - Familiarize yourself with the editing environment and Git
  - Build two types of distributed systems
- This is an **individual** assignment
  - You must write all your own code
  - You may discuss general ideas with other students and link them help documentation
  - You may give general advice for debugging and design, but you should **never** have your code open while looking at someone else's code!
  - This is more lenient than many classes, don't abuse it!

Why Go?

- Go has become a very popular language for building distributed systems
- Born at Google by Robert Griesemer, Rob Pike and Ken Thompson (C/Unix)
- Power and performance of C, but with the convenience and safety of more modern languages

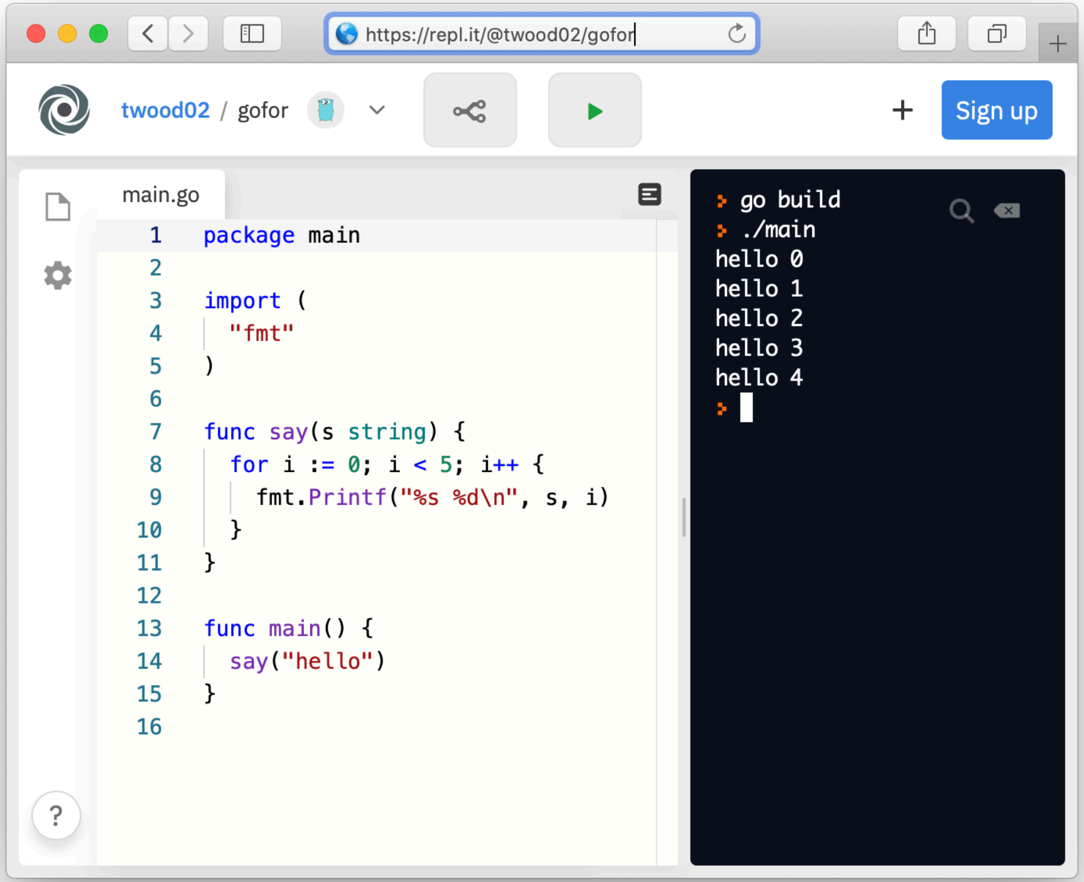
- Learn more: https://golang.org/doc/faq

"Go … [attempted] to combine the ease of programming of an interpreted, dynamically typed language with the efficiency and safety of a statically typed, compiled language. It also aimed to be modern, with support for networked and multicore computing."

Phase 1: Sequential Sum

- Starter code:
  - Reads a file and puts numbers in an array
- Your code:
  - Use a for loop and add up the numbers
  - Add command line parameter support
  - (this should be easy even if you've never touched go)

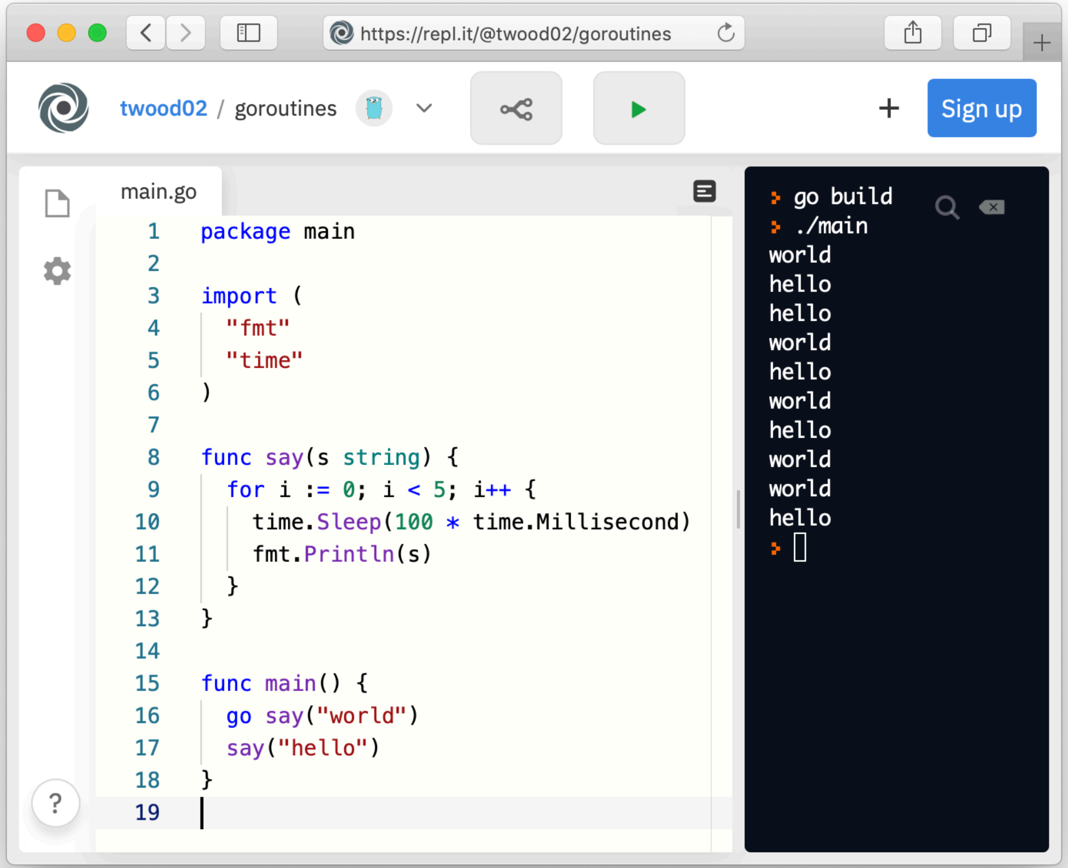
- Hint: Take a tour of Go
  - https://tour.golang.org/list



https://repl.it/@twood02/gofor

PHASE 2: PARALLEL SUM

- Main thread still reads in file and makes array (see starter code)
- Use Goroutines to parallelize the addition
  - A **Goroutine** is a lightweight thread
  - **What does this mean with regards to concurrency and parallelism?**
- How will the main thread and goroutines coordinate?
  - Need to pass numbers to be summed
  - Need to get back the result
  - Hint: learn about **Go Channels**!



https://repl.it/@twood02/goroutines

Phase 3: HTTP+RPC

- Let's make a "real" distributed system! Two Go programs:
- HTTP Frontend
  - Accepts a client request specifying file to process
- RPC Backend
  - Receives a Remote Procedure Call from frontend to trigger the summation
  - Uses goroutines to parallelize like in prior phase