CDS Lab

Summer Semester 2017

Goal of the lab

- Introduction to concurrency
- Gain experience in parallel programming
- Basic insight into technologies/tools such as Docker and web-services
- Evaluation of different approaches

General Information

- Franz Gregor, Franz.Gregor@tu-dresden.de
- Robert Krahn, Robert.Krahn@tu-dresden.de
- Please pose problem specific questions in our Auditorium group:

  “Lab: Concurrent and Distributed Systems”

  https://auditorium.inf.tu-dresden.de/en/groups/2023686

- Repository: https://bitbucket.org/r0bcrane/fcds-lab-2017

Introduction

Single-threaded code

- Underutilized hardware, (typically multicore machines)
- Not scalable

Concurrent code

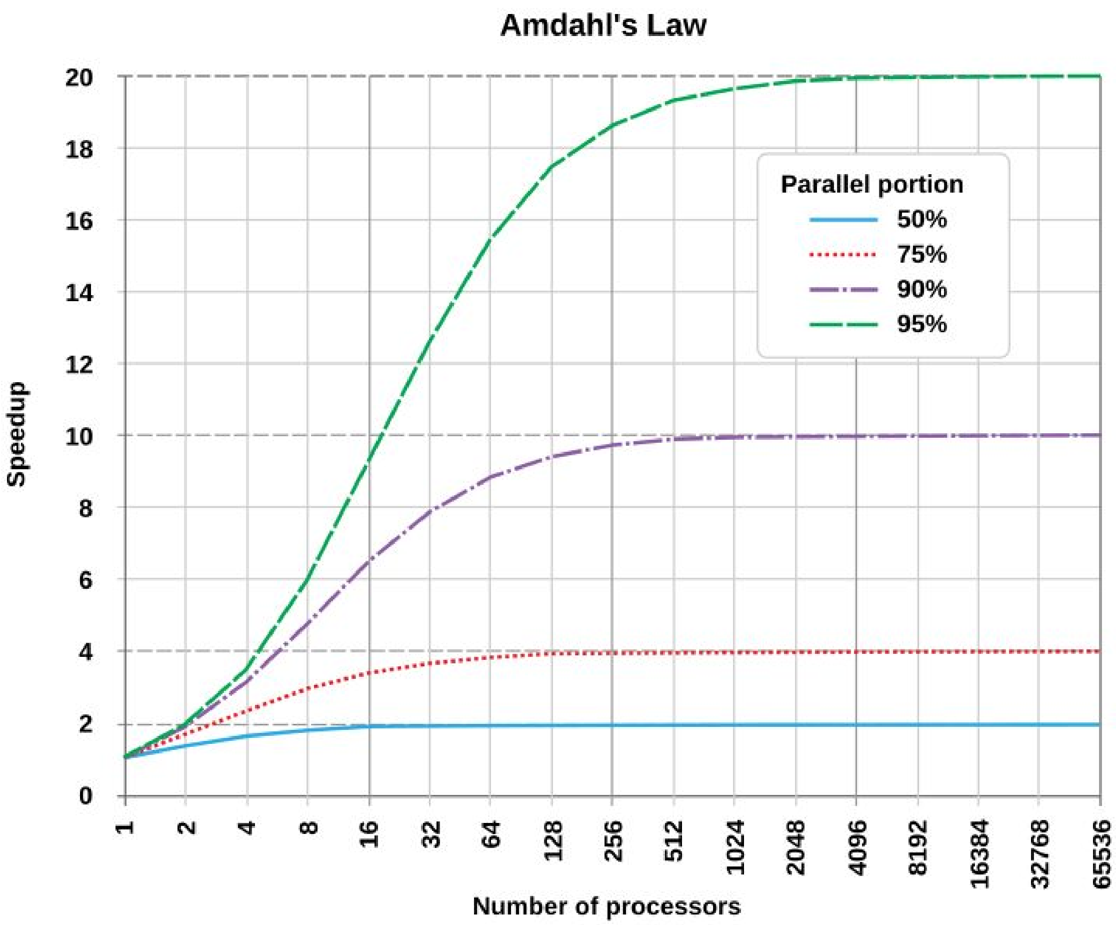
- Low-level concurrency using threads
- Higher level concurrency using fork/join model or actors
- Leverage multicore hardware

Amdahl’s Law

- Two parts to each code:
  - Parallelizable code
  - Fixed part, parallelization not possible

- Theoretical estimation of speedup through parallelization

$$S(N) = \frac{1}{(1-P)+\frac{P}{N}}$$



(https://en.wikipedia.org/wiki/Amdahl%27s_law)

Concurrency Concepts

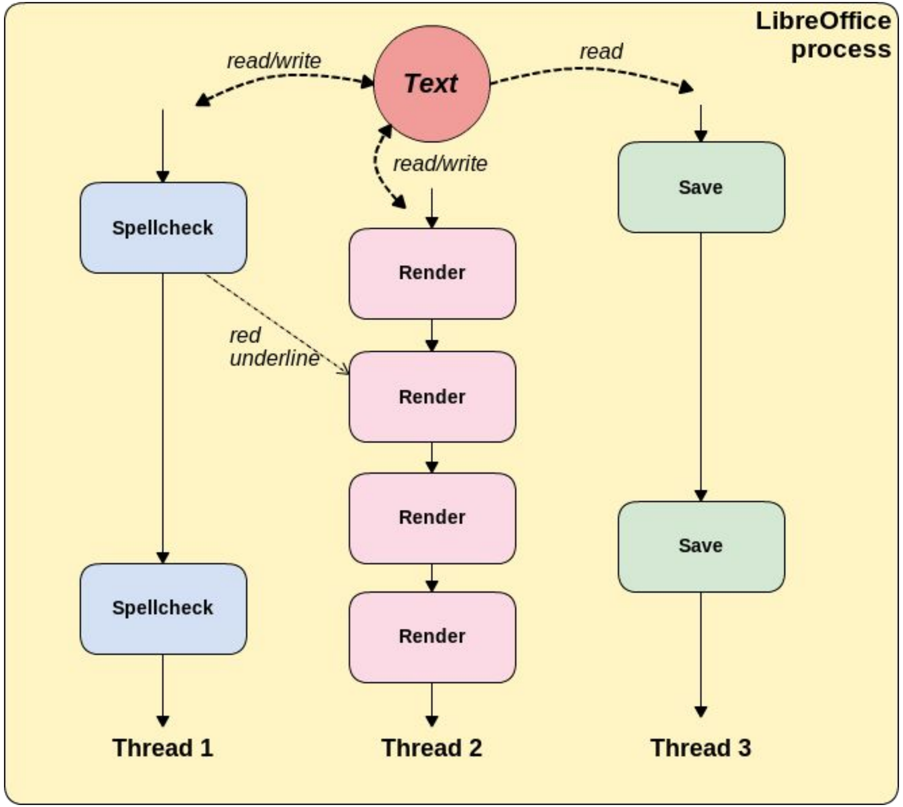
- Thread model
- Fork-Join model
- Message Passing model
- Actors model

- Code/Language/Library magic

Thread Model

- Shared memory model
- Single "heavy weight" process has multiple "light weight", concurrent execution paths (threads)
- Threads communicate via shared variables and/or sending signals
- Threads split the tasks
- Most control, least safety/comfort
- Issues: locks, semaphores

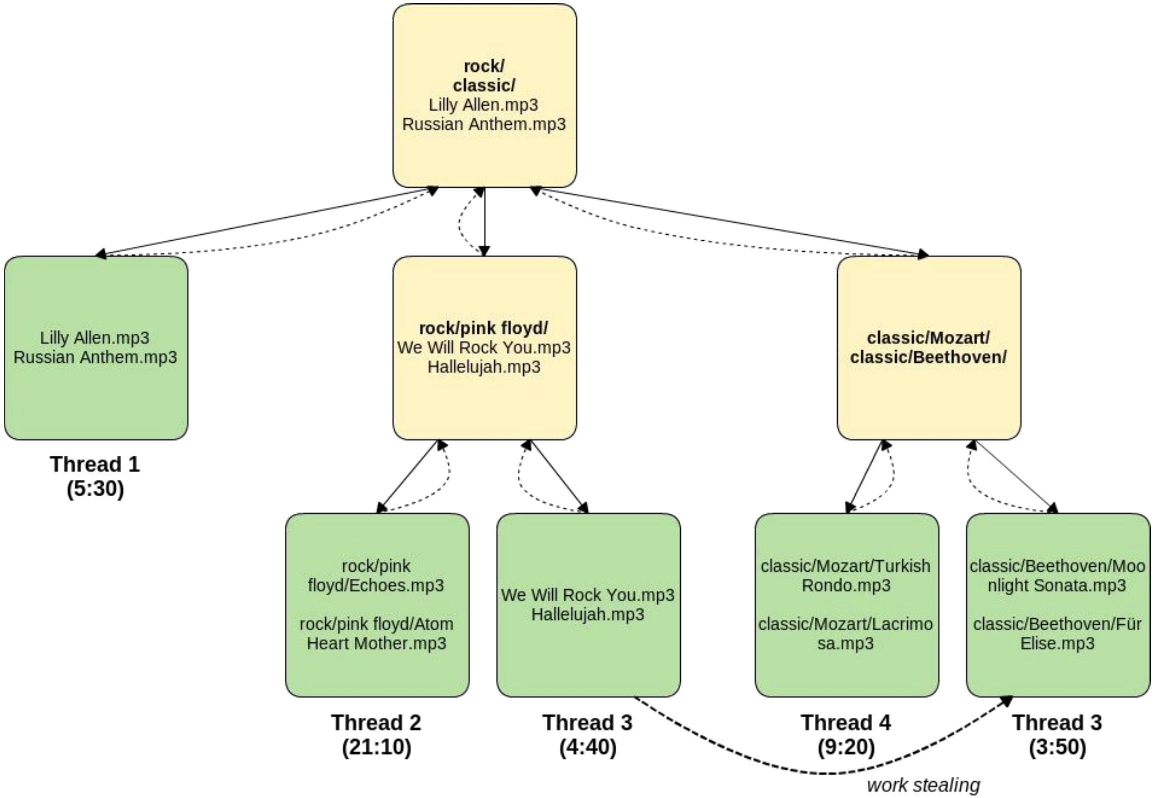
Thread Model



Fork-Join Model

- Divide & Conquer model to solve hierarchical problems
- Split a problem into smaller sub-problems and recursively apply the same algorithm to each sub-problem (Fork)
- Solutions of all sub-problems are combined to solve the initial problem (Join)
- Sub-problems do not share data: no locks, no races!

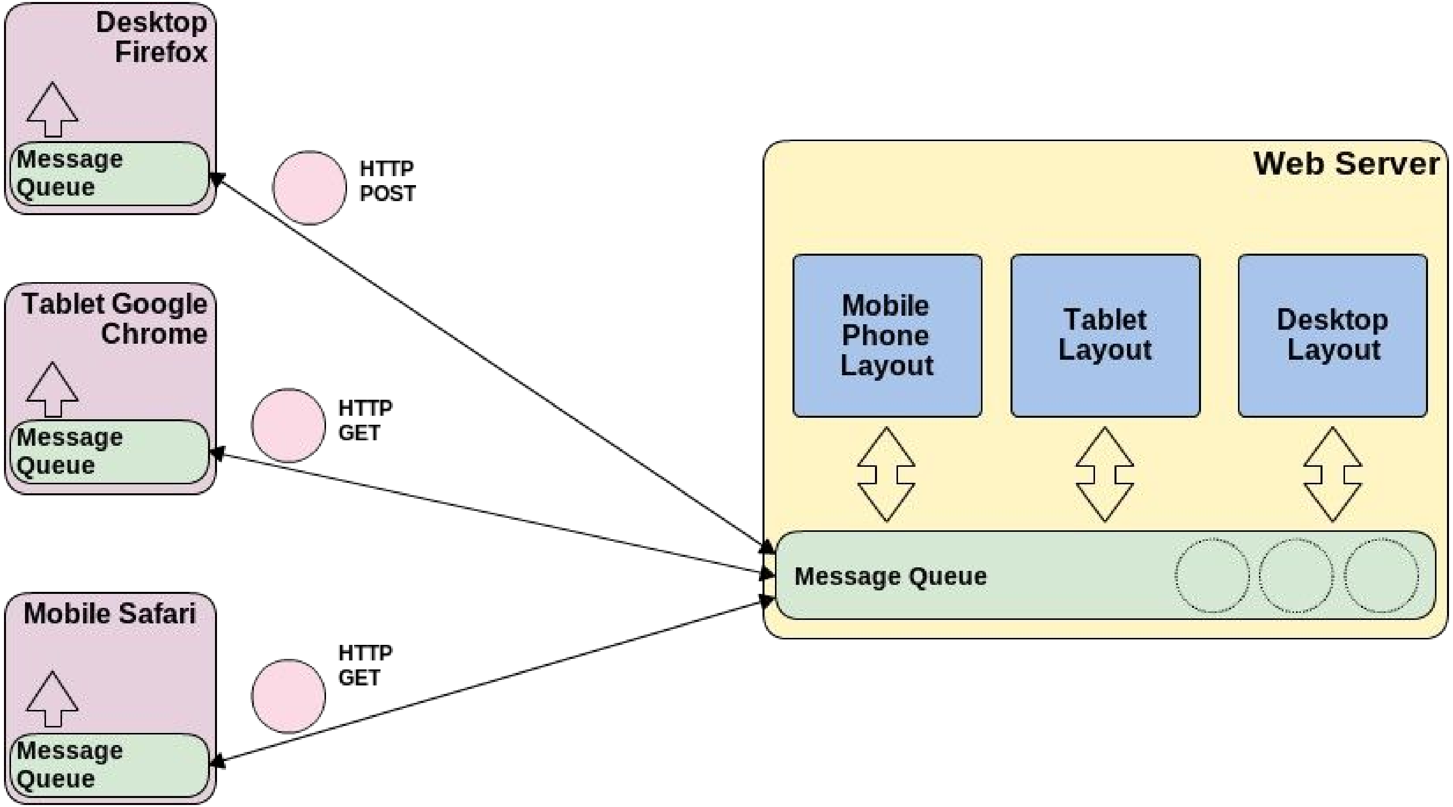
Fork-Join Model



Message Passing Model

- Different objects (actors, agents) communicate only via sending and receiving messages
- No shared data – messages contain full copies
- Need an infrastructure to communicate – channels (message queues, pipes, sockets)
- Synchronous or Asynchronous
- Great for distributed programming, useful for concurrent programming

Message Passing Model



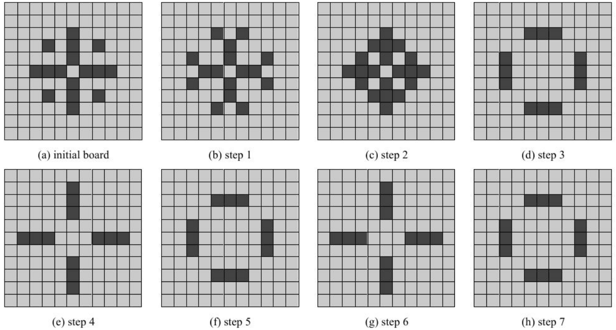
The Tasks

- Tasks provided by the 11th Marathon of Parallel Programming 2016
- https://bitbucket.org/r0bcrane/fcds-lab-2017/
- Sequential solution will be made available there
- Discuss issues at:

https://auditorium.inf.tu-dresden.de/en/groups/2023686

The Tasks

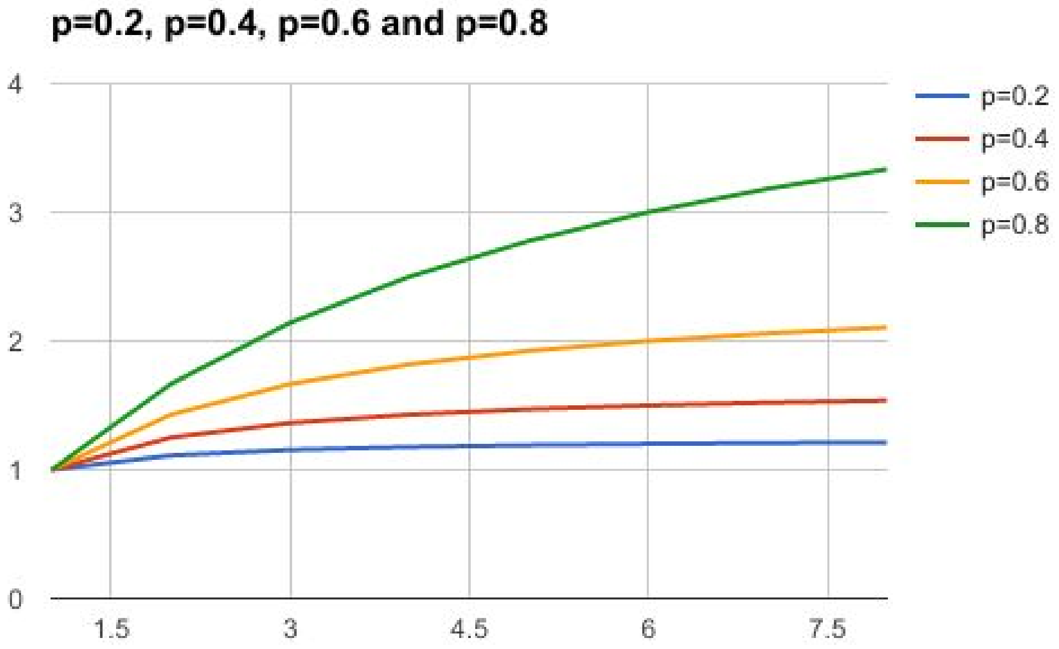
- String Parsing
  - Is a string part of a given language?
- Game of Life
- Sudokount
  - How many solutions exist for a given Sudoku puzzle?
- Color Histogram
  - Analyse an image for its color distribution.



How to gather points

- You earn points for each task
    - Points = speedup of your program with 8 cores over (our) sequential solution
- Total points sum of all four tasks
- Theoretical max speedup per task: 8
- Consult Amdahl’s law!
- We will execute your code!

Amdahl’s law, again



Credits

6 credits if:

- Work in a team of two and your team gathers 9 points
- Work alone and gather 7 points

3 credits if:

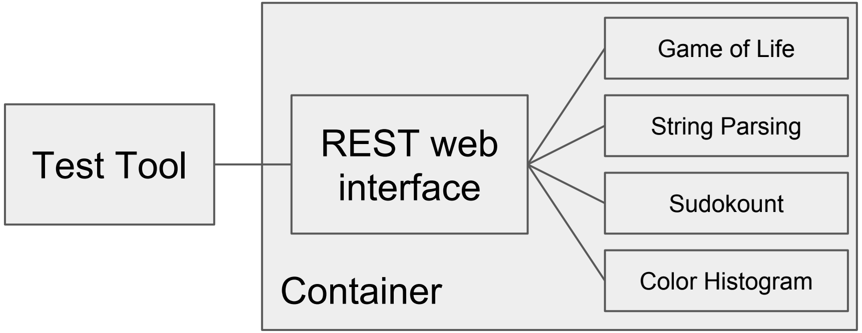
- Work alone and gather 5 points

Submission & Evaluation

- Create your own Git repository at bitbucket or github
- Make it accessible (read) for us
- Write us a registration e-mail
    - Name, partner, student id, repository
    - Until 7.5.17
- Any deliverables must include a revision number (commit hash)
- Your code will automatically be downloaded and tested
- Details will be announced soon.

Submission & Evaluation

- Use whatever language/framework you want
- Programs are built and evaluated in Docker containers
  - Checked for correct solutions
  - Measured for speedup with 8 cores with our test tool
- Find Docker template & test tool in our Git repository
- Use containers for development
  - Less hassle near the deadline
  - We measure the same thing as you



Dates

Intermediate presentation:

- Date: 22.05.2017 (time and room will be announced)
- Present the ideas and concepts at midterm
  - 5-slide presentation of current status (sketches of your system)

Final presentation:

- Deliverables deadline: 05.07.2017 / 0:00 am
  - Your solutions will automatically be evaluated
  - We’ll notify you about the results
- Presentation: 10.07.2017 (time and room will be announced)

Final Presentation

10 minutes to present your solutions, at least with the following subjects:

- a sketch of the architecture of your system
- a list of the (concurrent) data structures used
- depict the scalability points of your program
- plots showing the performance of your system in terms of throughput compared with the number of threads/processes used

Thank you

Please check regularly:

- Our web site,
- Git repository,
- Auditorium