

Vaja 4 - Potapljanje ladjic

Postavitev in upravljanje računalniških oblakov

David Rubin
(david.rubin@student.um.si)

16. november 2018

1 Opis naloge

Implementirati je bilo potrebno MPI program, ki izvaja večigralsko potaplanje ladjic. Slednje je izvedeno tako, da je neko vozlišče gospodar (v mojem primeru *trusty0*), ki hrani stanje igre in sprašuje igralce po njihovih potezah. To vozlišče skrbi tudi za spletni strežnik, ki omogoča vpogled v stanje med igranjem igre.

2 Opis rešitve

Rešitev sem implementiral kot Vagrant konfiguracijo gruče, skripto za kopiranje programa na vse instance in Python MPI program opisanega potapljanja ladjic. V Vagrant datoteki se namesti vse potrebno (*mpi4py*, *nginx* ipd.) v 10 instanc okolja Ubuntu 14.04 Trusty Tahr. Poskrbi tudi za povezljivost med instancami, kot tudi za fiksno določene IP naslove.

Skripta za kopiranje programa iz datoteke prebere IP naslove vseh 10 strežnikov in podan parameter (ime programa) prekopira na vsako izmed njih.

Program napisan v Pythonu je viden v kodnem bloku 1. Skratka, vzpostavi se igralno polje in kopija tega, ki drži pozicije ladjic, ki so fiksno določene. Potem glavno vozlišče pošlje polje vsem ostalim, ti pa odgovorijo z naključno izbranimi koordinatami, ki pa se preverijo, da so še neigrana polja. Ko se potopi zadnja ladjica, glavno vozlišče namesto polja vsem igralcem pošlje vrednost -1, ki zaključi igro. Na gospodarju se izpišejo rezultati igralcev in program se zaključi.

Listing 1: Program za večigralsko potaplanje ladjic

```
from mpi4py import MPI
from random import randint
from time import sleep
from copy import deepcopy
import sys

html_file = '/home/vagrant/www/index.html'
```

```

def buildHtml(pf, player, move, scores):
    """
    Zbuilda HTML iz igralnega okolja, igralca in poteze

    pf          ... 2d seznam igralne plosce
    player      ... trenutni igravec (ime procesorja)
    move        ... seznam z X in Y koordinato
    scores      ... seznam z rezultati vseh igralcev
    """

    html = """
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
            ↪ scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <meta http-equiv="refresh" content="1">
        <title>Vecigralsko potapljanje ladjic</title>
        <link href="https://fonts.googleapis.com/css?family=Open+
            ↪ Sans" rel="stylesheet">
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn
            ↪ .com/bootstrap/4.1.3/css/bootstrap.min.css"
            ↪ crossorigin="anonymous">
    <style>
        body {{
            background-color: #37474f;
            color: #fff;
        }}
        .container {{
            margin: auto;
            margin-top: 50px;
            font-family: 'Open Sans', sans-serif;
        }}
        table, td {{
            border: 1px solid black;
            border-collapse: collapse;
            background-color: cornflowerblue;
        }}
        tr, td {{
    """

```

```

        width: 30px;
        height: 25px;
        text-align: center;
        font-size: 12pt;
        font-weight: bolder
    }}
    #current_player, #played_move {{
        font-weight: bold;
    }}
    .miss {{
        color: #be1131;
    }}
    .miss:after {{
        content: "\\2716";
    }}
    .hit {{
        background-color: grey;
    }}
    .row {{
        margin-top: 50px;
    }}
    .score {{
        font-size: 18pt;
        margin-left: 20px;
    }}
</style>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-6">
                <table id="playing_field">
                    {}
                </table>
            </div>
            <div class="col">
                <h4>Scores:</h4>
                {}
            </div>
        </div>
    </div>
    <div class="row">

```

```

        <div class="col">
            <p>Trenutno na potezi: <span id="
                ↪ current_player">{}</span> <br
                ↪ />
            Odigrana poteza: <span id="
                ↪ played_move">{}, {}</span></p>
                ↪ >
        </div>
    </div>
</div>
</body>
</html>
"""

html_row = ""
for row in pf:
    html_row += "<tr>"
    for field in row:
        if field == -1:
            # Uporabnik je probal in nic ni
            html_row += "<td class='miss'></td>"
        elif field == 0:
            html_row += "<td></td>"
        else:
            html_row += "<td class='hit'>" + str(field)
            ↪ + "</td>"

    html_row += "</tr>"
# Zanimari master vozlišce (privzeto da je 0)
html_scores = ""
tmp = enumerate(scores)
scores_desc = sorted(tmp, key=lambda tmp:tmp[1], reverse=True)
for rank, score in scores_desc:
    if rank > 0:
        html_scores += "<p>trusty" + str(rank) + "<span_
            ↪ class='score'>" + str(score) + "</span></p>"
            ↪

with open(html_file, 'w') as f:
    f.write(html.format(html_row, html_scores, player, move[0],
        ↪ move[1]))

```

```
def createShips(pf):
```

```

"""
    Ustvari n ladjic velikosti size na igralno površino pf

    pf          ... igralno polje
    size ... velikost ladjice
    n          ... stevilo ladjic, ki se naj kreirajo
"""

ships = deepcopy(pf)
# Ladjice velikosti 5
for i in range(15, 20):
    ships[1][i] = 5
    ships[i][18] = 5
# Ladjice velikost 4
for i in range(8, 12):
    ships[4][i] = 4
    ships[i][2] = 4
# Ladjice velikosti 3
for i in range(6, 9):
    ships[11][i] = 3
    ships[i][19] = 3
# Ladjice velikost 2
for i in range(13, 15):
    ships[17][i] = 2
    ships[i][7] = 2
# Ladjice velikosti 1
ships[0][0] = 1
ships[5][5] = 1
return ships

# Osnovni parametri MPI programa
comm = MPI.COMM_WORLD
size = MPI.COMM_WORLD.Get_size()
rank = MPI.COMM_WORLD.Get_rank()
name = MPI.Get_processor_name()
master = 0
# Ustvari 20 X 20 igralno polje
play_size = 20
playing_field = [[0 for _ in range(play_size)] for _ in range(play_size)]
# Ustvari ladjice. Trenutno deluje le tako, da so ladje ob vsaki ifri
    ↪ fiksno dolocene

```

```

ships = createShips(playing_field)
# Polje z rezultati igralcev. Ce igralec zadane ladjo dobi 1 tocko ne glede
    ↪ na velikost
# ladje. Skupno stevilo tock je lahko največ  $5*2 + 4*2 + 3*2 + 2*2 + 2*1 =$ 
    ↪ 30
scores = [0 for _ in range(size)]
# Prikazi zacetno igralno polje
if rank == master:
    buildHtml(playing_field, name, [0,0], scores)
# Igraj dokler so nepotopljene ladje
ships_standing = True
while ships_standing:
    # Rank 0 predstavlja nadzornika igre, skrbi za stanje in posilja
        ↪ ostalim igralcem
    # trenutno stanje in od njih pricakuje potezo. Prav tako vodi
        ↪ evidenco o rezultatu
    # in posodablja stanje datoteke za spletni streznik
    if rank == master:
        # Z BeautifulSoup kreiraj preberi HTML datoteko in vstavi
            ↪ igralno polje
        # Po vrsti sprasuj ostale igralce
        for node in range(1,size):
            # Poslji igralcu trenutno stanje igralne plosce
            comm.send(playing_field, dest=node, tag=11)
            # Sprejmi od igralca njegovo potezo
            move = comm.recv(source=node, tag=11)
            x = move[0]
            y = move[1]
            # Preveri ali je igralec zadel
            if (playing_field[x][y] == 0 and ships[x][y] > 0):
                # Privzemi, da je zadnji del ladje potoplen
                ships_standing = False
                # Igralec je dobil nov zadetek! Povej mu
                    ↪ rezultat in osvezi polje
                scores[node] += 1
                # Povej katera ladja je na zadetku
                playing_field[x][y] = ships[x][y]
                # Zbrisi kos ladje iz rezultatov in preveri
                    ↪ , ce je se kaj ostalo
                ships[x][y] = 0
            for row in ships:

```

```

        if sum(row) > 0:
            ships_standing = True
            break
    elif playing_field[x][y] > 0:
        # Igralec je poskusal utrofiti ze zadeto
        ↪ polje
        pass
    else:
        # Podaj da je polje prazno
        playing_field[x][y] = -1
        # Izvajaj korake na vsako sekundo (da cas HTML da
        ↪ se prikaze)
        sleep(0.1)
        # Posobodi HTML
        buildHtml(playing_field, 'trusty'+str(node), move,
        ↪ scores)
        # Prislo je do konca in master je zakljucu igro, obvesti
        ↪ igralce, da
        # lahko prenehajo z igranjem
        if not ships_standing:
            buildHtml(playing_field, 'Konec_igre', [-1, -1],
            ↪ scores)
            print("All_ships_have_fallen")
            print("Final_scores:")
            for node in range(1, size):
                print("\tNode{}: {}".format(node, scores[
                ↪ node]))
                comm.send(-1, dest=node, tag=11)
        # Ostali clani v clustru so igralci, vsak poskusa potopiti ladjico
        ↪ z neko potezo,
        # ki jo poslje gospodarju
        else:
            # Pridobi zadnjo igralno ploskvo od igralca
            playing_field = comm.recv(source=master, tag=11)
            if playing_field == -1:
                # Prisel je signal za zakljucek igre, vse ladje so
                ↪ potopljene
                break
            # Nacrtuj naslednjo potezo. Pri tem ne poskusaj zadeti
            ↪ polja,
            # ki je ze bilo odigrano (!= 0)

```

```

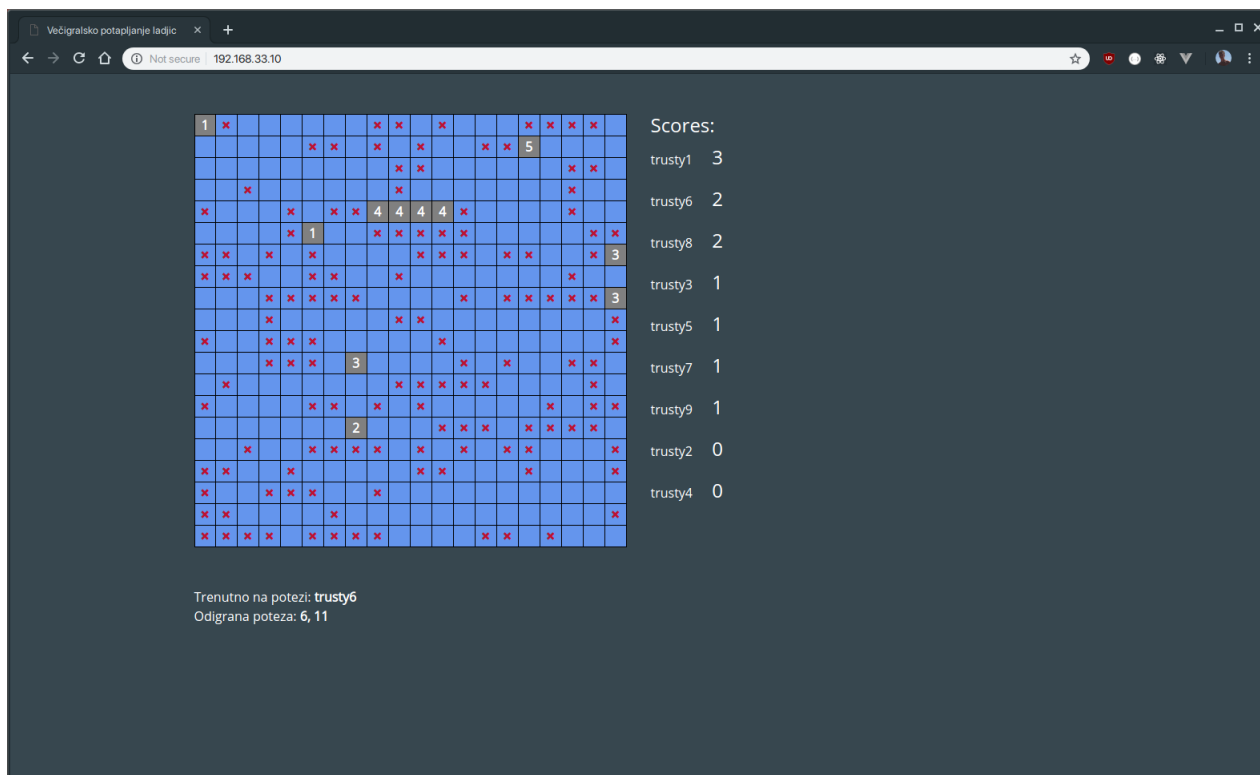
my_moveX = randint(0, len(playing_field[0])-1)
my_moveY = randint(0, len(playing_field)-1)
while playing_field[my_moveY][my_moveX] != 0:
    #print('Tryin move {}, {}'.format(my_moveX,
    ↪ my_moveY))
    # Novo polje poracunaj kar po vrsti (naprej od
    ↪ nakljucnega)
    my_moveX = (my_moveX + 1) % len(playing_field[0])
    if my_moveX == 0:
        my_moveY = (my_moveY + 1) % len(
            ↪ playing_field)
    # Svojo potezo poslji kot seznam, kjer je na prvem mestu Y,
    # na drugem pa X koordinata poteze
    my_move = [my_moveY, my_moveX]
    # Poslji svojo potezo gospodarju igre
    comm.send(my_move, dest=master, tag=11)

```

Med kodo lahko opazimo tudi HTML, na gospodarju imamo namreč nameščen spletni strežnik nginx, kateremu pa smo spremenili konfiguracijo, da streže datoteko /www/index.html. Vsebino te datoteke spreminjamo z zgoraj navedenim programom. Na sliki 1 vidimo spletni vmesnik programa med izvajanjem igre, na sliki 2 je spletni vmesnik po končani igri, na sliki 3 pa je viden še ukaz, ki je pognal igro.

3 Izjava o izdelavi domače naloge

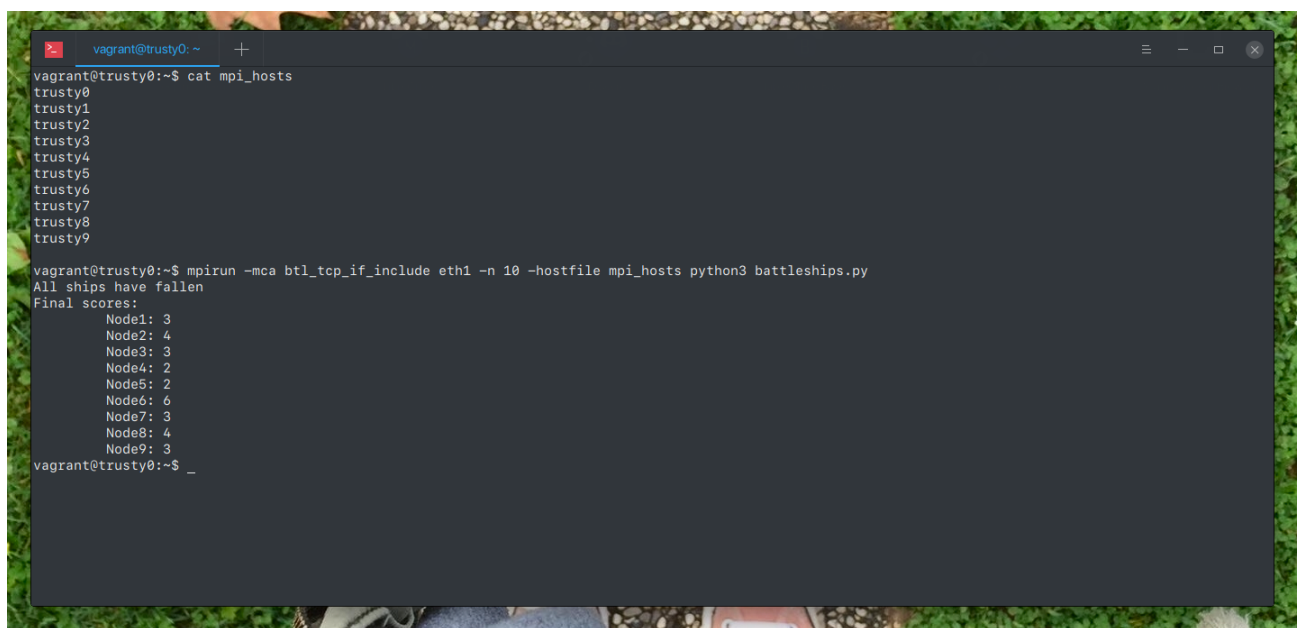
Domačo nalogo in pripadajoče programe sem izdelal sam.



Slika 1: Spletni vmesnik med igranjem igre



Slika 2: Spletni vmesnik po končani igri



```
vagrant@trusty0: ~  
vagrant@trusty0:~$ cat mpi_hosts  
trusty0  
trusty1  
trusty2  
trusty3  
trusty4  
trusty5  
trusty6  
trusty7  
trusty8  
trusty9  
  
vagrant@trusty0:~$ mpirun -mca btl_tcp_if_include eth1 -n 10 -hostfile mpi_hosts python3 battleships.py  
All ships have fallen  
Final scores:  
Node1: 3  
Node2: 4  
Node3: 3  
Node4: 2  
Node5: 2  
Node6: 6  
Node7: 3  
Node8: 4  
Node9: 3  
vagrant@trusty0:~$ _
```

Slika 3: Zaključen ukaz, ki je pognal zgoraj prikazano igro