Paralelizacija grafovskih algoritmov v funkcijskih jezikih

Avtor: Tjaž Eržen, Mentor: doc. dr. Matija Pretnar

Kratka predstavitev pri Diplomskem seminarju, 12. december 2022

Uvod

1 Zakaj porazdeljeni sistemi?

2 OCaml

3 Moja diplomska naloga

Uvod

- Hitro ni dovolj hitro?
  - Današnji računalniki so 100-krat hitrejši od tistih izpred desetletja, vendar sama računska moč pogosto ni dovolj.
  - Današnji računalniki imajo bistveno več spominskih kapacitet od tistih izpred desetletja, vendar sama kapaciteta spomina pogosto ni dovolj.

Uvod

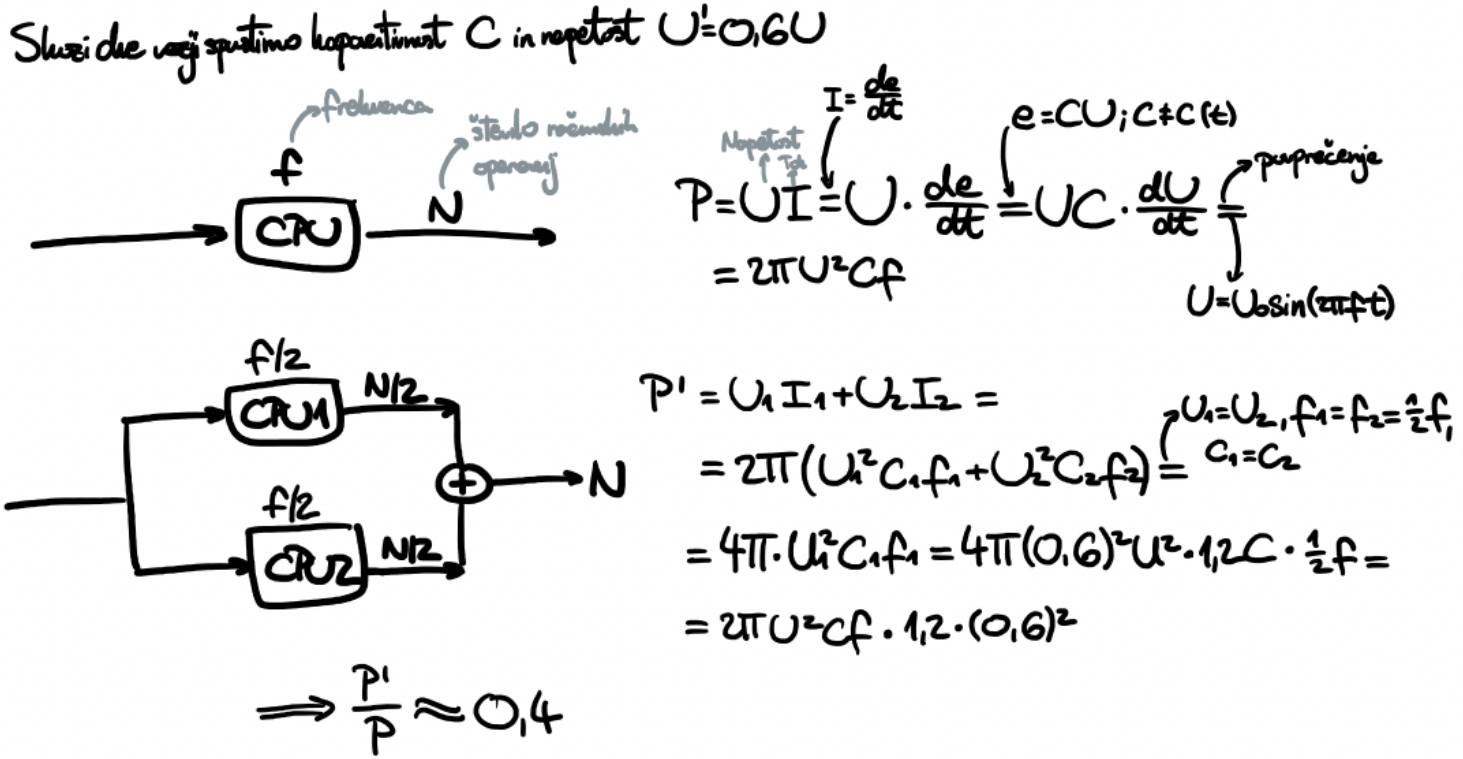
- Veliki izzivi “large-scale” sistemov:
  - Računalniško-matematični problemi (npr. množenje matrik).
  - Podatkovna analitika (večinoma gre za paralelizacijo na grafičnih karticah)
  - Modeliranje vremena in obnašanje okolja
  - Računsko težki računski problemi v fiziki in biologiji
  - ...

Je vzporedno programiranje sistemov res neizogibno?

- Zmogljivejše sisteme trenutno dobimo predvsem z množenjem gradnikov
- Tehnologija izdelave čipov počasi zadeva ob oviro - bo Moorov zakon zdržal?
  - Do leta 2005 50% povečanje tranzistorjev vsako leto (60x v 10 letih)
  - Danes 20% povečanje tranzistorjev vsako leto (6x v 10 letih)
- Nastane potreba po drugačnem načinu optimizacije

Je vzporedno programiranje sistemov res neizogibno?

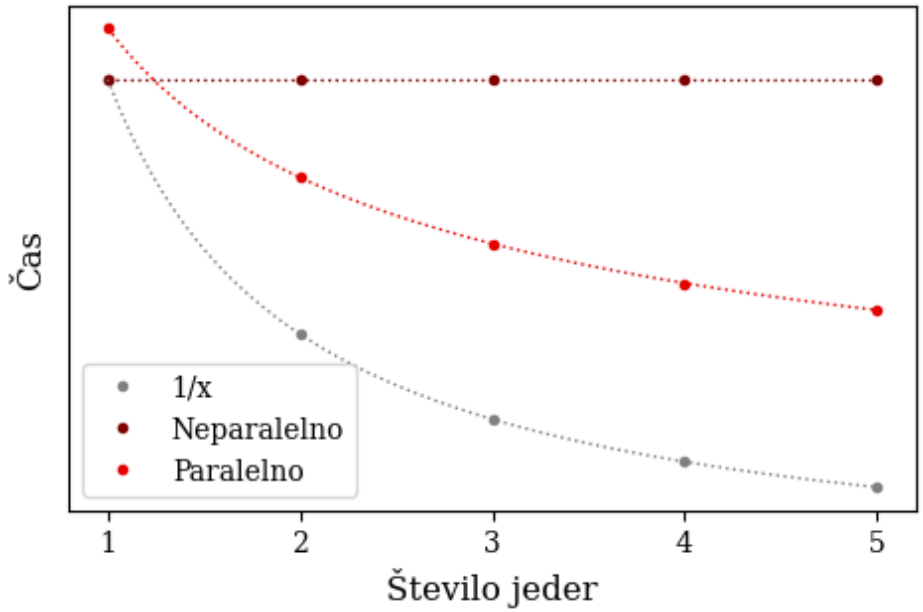
Slika: Fizikalno - večje oddajanje toplote v monolitnih kot v porazdeljenih sistemih



Cilj paralelnega programa

Nekatere podatkovne strukture se da paralelizirati, medtem ko se drugih ne.

Slika: Paralelni programi so z večanjem stevila jeder hitrejši.



OCaml

- Funkcijski jezik
- Letošnja različica verzije *5.0.0*, ki (med drugim) vpelje knjižnico za delo z vzporednimi procesi.

Slika: Paralelizacija računanja Fibonaccijevih števil v OCamlu

```
let n = try int_of_string Sys.argv.(1) with _ -> 40

let rec fib n = if n < 2 then 1 else fib (n - 1) + fib (n - 2)

let main () =
  let d1 = Domain.spawn (fun _ -> fib n) in
  let d2 = Domain.spawn (fun _ -> fib n) in
  let r1 = Domain.join d1 in
  Printf.printf "fib(%d) = %d\n%!" n r1;
  let r2 = Domain.join d2 in
  Printf.printf "fib(%d) = %d\n%!" n r2

let _ = main ()
```

Zakaj pisati vzporedne programe?

- Učinkovitost
- Dobra novica: Potrebno znanje paralelizacije računalniških sistemov
- Slaba novica: Razvoj prevajalnikov, ki bi znali naše zaporedne programe samostojno pretvoriti v vzporedno obliko. Ideja: Iščejo se tipični programski konstrukti, ki se jih pretvarja v paralelno obliko

Cilji diplomske naloge

- Paralelizacija računalniškega procesorja v funkcijskem jeziku OCaml
- Se spoznati s paralelizacijo funkcijskih jezikov
- Pregled že obstoječih paralelnih knjižnic v drugih funkcijskih jezikih (npr. Haskell) in implementacija le-teh v OCamlu
- Analiza podatkovnih struktur in algoritmov, primernih za paralelizacijo

Začetni plan

- Implementacija osnovnih matematičnih funkcij z “naivno” paralelizacijo (*map, apply, sum, ...*)
- Ugotoviti, kdaj se nekaj splača paralelizirati in kdaj ne
- Implementacija/ razvoj manj trivialnih paralelnih algoritmov (grafi)