Paralelno programiranje

Vežba 1 – Uvod u paralelno programiranje

Otkrivanje pogodnih mesta u kodu za paralelnu obradu 1/4

- Razlike između konkurentnih i paralelnih sistema:
  - Konkurentni sistemi – više zadataka koji počinju, izvršavaju se (ne nužno u kontinuitetu) i završavaju tako da postoji preklapanje
  - Paralelni sistemi – više zadataka se istovremeno izvršava na istom hardveru koji sadrži više računarskih resursa kao što su procesori sa više jezgara, procesor + grafička kartica itd…
- 3 osnovna pitanja koja postavljamo:
  - Gde uvesti paralelizam (koji to segment koda oduzima najviše vremena)?
  - Kako uvesti paralelizam (koje tehnike, koji šabloni, alati, okviri)?
  - Postoji li problem sa sinhronizacijom?

Otkrivanje pogodnih mesta u kodu za paralelnu obradu 2/4

- Određivanje mesta za paralelizaciju:
  - Uz pomoć alata namenjenih za to kao što je Intel Advisor (potreban je ceo paket koji uključuje i Intelov prevodioc)
  - „Peške“ uvidom u kod i merenjem vremena
- Najbolji kandidati za paralelizaciju su uglavnom petlje i rekurzivne funkcije, ali sve što oduzima puno vremena se treba uzeti u razmatranje

Otkrivanje pogodnih mesta u kodu za paralelnu obradu 3/4

- Kako uvesti paralelizam?
- Postoje razne metode i okviri, svaki sa svojim prednostima i manama:
  - Niti (daju programeru najveću kontrolu, ali iziskuju najveću odgovornost i poznavanje arhitekture)
  - Programski okviri kao što su OpenMP, Cilk++, **Intel OneTBB** (pojednostavljeno uvođenje paralelizma, ali ograničene mogućnosti)
  - Okviri za heterogene sisteme kao što je OpenCL (veća kompleksnost)

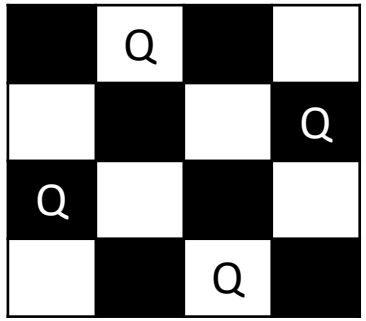
- Mi ćemo se na ovom predmetu fokusirati na OneTBB veći deo semestra, a manji deo ćemo raditi OpenCL

Otkrivanje pogodnih mesta u kodu za paralelnu obradu 4/4

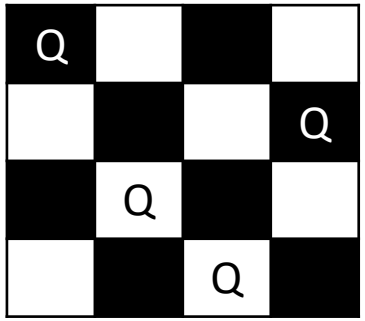
- Sinhronizacija – problem koji ne postoji u serijskim programima
- Šta se dešava ukoliko više niti / zadataka pristupa istoj promenljivoj / strukturi / klasi?
- Kako obezbediti konzistentnost?
- Imamo li dovoljno resursa na raspolaganju tako da svaka nit / zadatak radi sa svojom lokalnom kopijom?
- Ovo su neka od pitanja na koje ćemo dati odgovor tokom semestra…

Primer N kraljica 1/3

- Za datu šahovsku tablu proizvoljne veličine, koliko se kraljica može postaviti tako da se međusobno „ne napadaju“



OK



Nije

- Za male dimenzije se problem može rešiti „silom“ tj. prostim listanjem svih mogućih kombinacija i odabirom ispravnih
- Problem je eksponencijalni i povećanjem dimenzija on postaje nerešiv u prihvatljivom vremenskom intervalu

Primer N kraljica 2/3

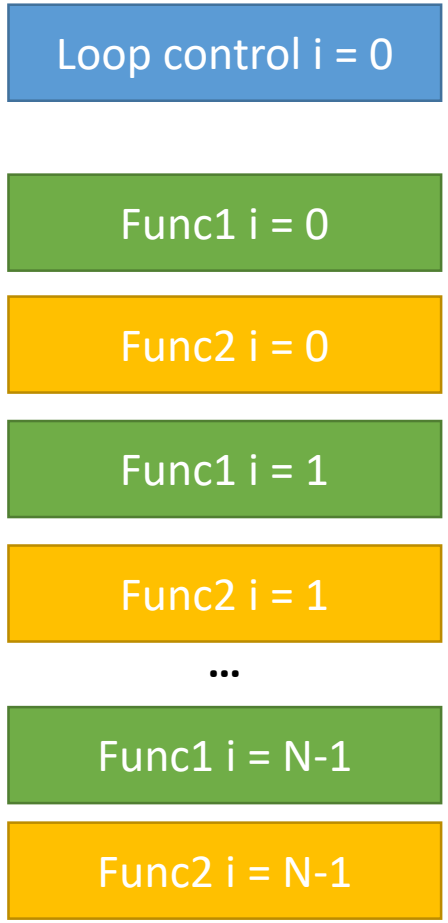
- Zato se koristi ***backtrack-ing*** algoritam koji radi na sledećem principu:
  - Postavi kraljicu na prvo slobodno polje
  - Postavi novu kraljicu u sledećoj koloni na prvo slobodno polje
  - Ako se kraljice napadaju, obriši poslednje dodatu kraljicu i postavi je na sledeće polje
  - Ako nema više slobodnih polja, obriši i prethodnu kraljicu i pomeri je na sledeće slobodno polje
  - Ako smo uspešno stavili kraljicu i u poslednju kolonu, našli smo jedno rešenje i treba ga zabeležiti
- Ovaj algoritam značajno smanjuje broj slučajeva koje treba obraditi
- Pored ovog algoritma, možemo se poslužiti i činjenicom da je tabla simetrična tako da za određene dimenzije možemo koristiti rešenja koja su „u ogledalu“ ili rotirana za 90 stepeni

Primer N kraljica 3/3

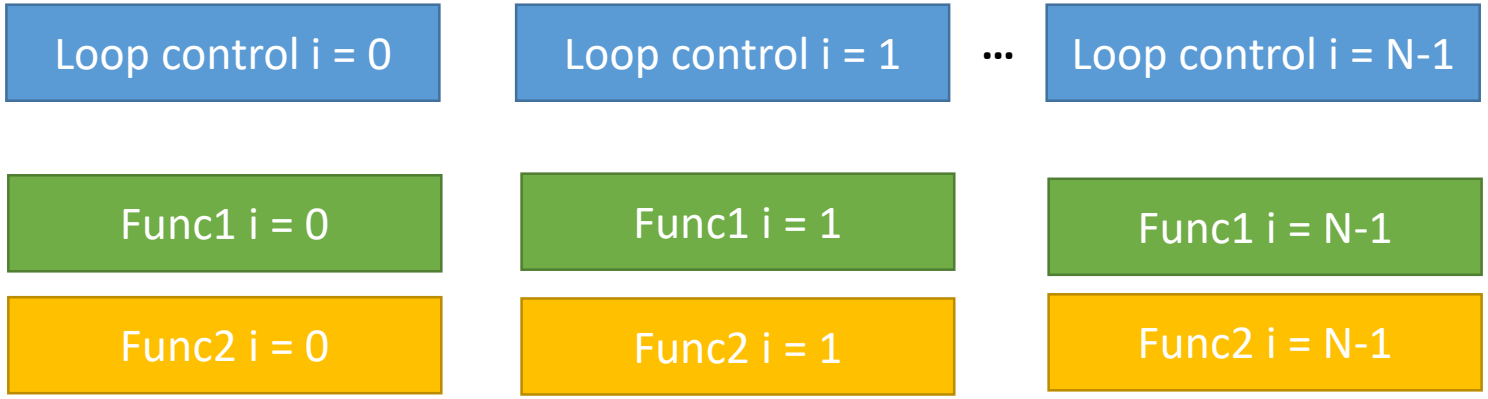
- Uvidom u kod i merenjem vremena se da zaključiti da se najviše vremena troši u petlji u kojoj se poziva funkcija setQueen()
- Unutar funkcije postoji petlja koja ide po svim vrstama i kolonama
- Da li možemo paralelizovati petlju?
- Da li su iteracije nezavisne?
- Odgovor na ova pitanja će nam dati odgovor na koji način možemo uraditi paralelizaciju.

Način paralelizacije petlji 1/2

- Nezavisne iteracije



Zavisne iteracije



Način paralelizacije petlji 2/2

- Ako su iteracije nezavisne, pribegavamo nekom od šablona za paralelizaciju petlji (OpenMP i OneTBB imaju parallel for, Cilk+ ima cilk_for)
  - Vrlo jednostavno rešenje, uglavnom dovoljno učinkovito
- Ako iteracije nisu nezavisne, onda koristimo neki vid podele posla u zadatke koje onda međusobno povezujemo i sinhronizujemo (zadaci iz K-te iteracije se moraju završiti pre nego što zadaci (K+1)-te iteracije krenu)

- Kakav je slučaj u zadatku N kraljica?

Kritične sekcije programa (1/2)

- Kritična sekcija programa predstavlja instrukciju ili blok instrukcija koji se mora izvršiti neometano
- Drugim rečima, ako jedna nit / zadatak izvršava dati blok, druge niti / zadaci **NE SMEJU** istovremeno izvršavati taj blok
- Kritične sekcije su vezane za međuzavisnost podataka i nastaju kada više niti / zadataka pristupa istom resursu od kojih barem jedna upisuje nešto (tj. menja vrednost)
- Kritične sekcije predstavljaju jedno veliko ograničenje koje utiče na nivo paralelizma koji se može postići i samim tim na maksimalno ubrzanje koje možemo postići paralelizacijom programa

Kritične sekcije programa (2/2)

- Kako rešiti kritične sekcije?
  - Zaključavanjem – najjednostavniji metod ali i najviše usporava program; prihvatljiv ako je kritična sekcija kratka (mali deo koda) i ako se zaključavanje neće događati jako često
  - Kopiranjem – svaka nit / zadatak će dobiti svoju kopiju kritičnog resursa s kojom će raditi, i potrebno je na kraju obezbediti mehanizam kako će se sve kopije spojiti u jedinstvenu vrednost; zahteva potencijalno puno resursa i ponekad jednostavno nije moguće
  - Menjanjem koda tako da se kritična sekcija izbegne – refaktorisanje koda i menjanje algoritma tako da ne postoji kritična sekcija

Intel OneTBB

- TBB (engl. Threading Building Blocks) je C++ okvir koji od korisnika prikriva detalje rada sa nitima
- Radi se na višem nivou apstrakcije (razmišlja se na nivou zadataka koje je potrebno izvršiti) koristeći biblioteke i šablone koji nam stoje na raspolaganju, a raspoređivaču zadataka se prepušta da sam kreira niti i rasporedi im zadatke
- Tokom semestra ćemo se detaljno upoznati sa nekim od fukncionalnosti koje ovaj okvir nudi

Paralelno rešenje N kraljica koristeći TBB

- spin_mutex nam omogućava da zaključamo promenljivu prilikom upisa u deljenu promenljivu
- parallel_for šablon za paralelnu for petlju nam omogućava da istovremeno pokrenemo veliki broj zadataka koji će nezavisno jedan od drugog da traže rešenje (tako što će krenuti od različite početne pozicije)
- Pokrenite oba primera i uporedite vreme izvršavanja i rezultate

Pokretanje vežbi

- Za previđenje i pokretanje programa ćemo koristiti alat Waf i Linux operativni sistem
- Koraci za izvršavanje:
  - Pozicionirati se u direktorijum gde se nalaze izvorne datoteke i waf skripta
  - Dodati skripti prava izvršavanja komandom chmod 777 waf
  - U terminalu kucati komandu ./waf configure
  - Ukoliko je prethodni korak uspešan kucati komandu ./waf build
  - Potom kucati komandu ./waf run –app=ime_aplikacije
- Za studente koji žele da rade na svojim laptop računarima na windows-u, najbolje je instalirati MS Visual Studio i uz njega OneTBB, napraviti novi program i samo ubaciti datoteke iz vežbe
- Oni koji insistiraju da koriste Windows a ne žele MS Visual studio, mogu koristiti bilo koje drugo okruženje ali moraju sami podesiti putanje do TBB-ovih biblioteka