0. Uvod o paralelizmu

U ranim fazama razvoja računarstva, pojavile su se ideje da se obrada na računarima paralelizuje. Prvi koraci u tom pravcu su pojava procesa i konkurentnih programa. Kroz eksplicitno izražavanje paralelizma i sinhronizaciju i komunikaciju procesa u konkurentnim programima, počelo se razmišljati o bržem izvršavanju, tako što bi se procesi izvršavali na različitim procesorima. Rane faze istraživanja i razvoja oblasti su se kretale u pravcu ostvarivanja pouzdanih i efikasnih mehanizama za realizaciju konkurentnih programa čije se izvršavanje može obaviti na proizvoljnom tipu mašine.

Taj pravac evolucije paralelizma doveo je do razvoja nekoliko podoblasti. Pre svega, po osnovnom mehanizmu sinhronizacije i komunikacije, mašine i primitive su podeljene na one sa deljenom memorijom i sa prosleđivanjem poruka. Oblasti distribuiranog i konkurentnog programiranja su se razvile, a pritom je interakcija između procesa ostvarivana preko sistemskih poziva operativnog sistema ili na nivou jezika za konkurentno programiranje, koji može opet da se osloni na operativni sistem.

Komunikacija preko deljene memorije je brža, ali je skalabilnost takvih rešenja mala, jer je sa brojem procesora rastao i broj potrebnih pristupnih portova na deljenoj memoriji. Da bi se ostvarila skalabilnost, bilo je neophodno napraviti mašine kod kojih procesori komuniciraju preko poruka, tako da mogu da imaju stotine procesora povezanih u kompleksnu računarsku mrežu. Osnovni problem takvih mašina bilo je kašnjenje u komunikaciji nastalo rutiranjem poruka kroz računarsku mrežu koja povezuje čvorove sa procesorima i memorijom.

Tako su počele da se razvijaju «mreže u kutiji» koje su imale specijalnu pravilnu topologiju mreže, prilagođenu tako da se moglo hardverizovati rutiranje u cilju minimizacije komunikacionog kašnjenja. Najpoznatiji sistemi sa pravilnom topologijom mreže su hiperkocke, dvodimenzione i višedimenzione rešetke i mreže sa topologijom zvezdastog grafa zasnovanog na permutacionim grupama {MJ 94A}{MJ 94}{JM 94}{GJ 06}. Tako su napravljeni superračunari koji su podržavali eksplicitan paralelizam na nivou procesa. Danas su mrežni uređaji u lokalnim računarskim mrežama dostigli tako velike brzine rutiranja i minimizirali kašnjenja, tako da više nisu neophodne specijalne topologije mreža. Veliki klasteri sa hiljadama računara povezanih u standardnu brzu mrežu predstavljaju oslonac za najkompleksnije simulacije i izračunavanja u svim oblastima nauke. Razvijen je GRID middleware koji na globalnom nivou omogućava kontrolisano i lako korišćenje takvih superračunarskih klastera na svetskoj Internet mreži {MJ 06}.

U toj oblasti neophodno je da se obezbedi balansiranje opterećenja procesora u toku izvršavanja paralelizovanog (konkurentnog odnosno distribuiranog) programa, kako bi se postigla veća brzina izvršavanja {MJ 99}{JS 01}. Podela na procese u postupku paralelizacije se najčešće radi u toku prevođenja. Međutim, optimalno preslikavanje procesa na procesore i određivanje redosleda izvršavanja u vreme prevođenja pokazali su se kao gotovo nerešivi problemi. Razlog za to su nepoznata vremena izvršavanja delova procesa zbog nepoznatih dinamičkih tragova izvršavanja u vreme prevođenja. Jednostavnije postavljeno, ulazni podaci koji su poznati tek u vreme izvršavanja definišu trajanje vremena izvršavanja

dinamičkih tragova svakog od procesa. Osim toga, i vreme komunikacije se ne može egzaktno odrediti u vreme prevođenja {MJ 99A}.

Ovaj nivo paralelizma naziva se nivoom grube i srednje granularnosti, ili paralelizmom na nivou procesa i niti {GAS 90}. Njime se često obezbeđuje i umanjena osetljivost na otkaze. Nivoi paralelizma grube i srednje granularnosti nisu predmet monografije, ali mnogi problemi paralelizacije na instrukcijskom nivou zajednički su sa problemima paralelizacije na nivou procesa.

Istovremeno sa razvojem paralelizma na nivou procesa, počeo je da se razvija i paralelizam na instrukcijskom nivou, odnosno paralelizam unutar procesa. To je prvo bilo inicirano razvojem protočnosti u hardveru, da bi danas dostiglo neslućene razmere. Osnovna razlika između paralelizma na instrukcijskom nivou i paralelizma na nivou procesa potiče od razlike u kom trenutku je poznato vreme trajanja osnovnih elemenata granularnosti kôda. Već smo rekli da je na nivou procesa taj element deo dinamičkog traga između dve komunikacione ili sinhronizacione primitive konkurentnog programa. Njegovo trajanje zavisi od skupa ulaznih podataka koji se razlikuju pri izvršavanju. Za razliku od toga, trajanje pojedinih instrukcija (operacija), kada posmatramo instrukcijski nivo paralelizma, poznato je u vreme prevođenja i izražava se u broju ciklusa procesora. Zato je kod instrukcijskog nivoa paralelizma lakše uraditi planiranje izvršavanja u vreme prevođenja (statičko raspoređivanje).

Poznata vremena izvršavanja instrukcija otvaraju neslućene mogućnosti paralelizacije programa u vreme prevođenja, ali olakšavaju paralelizaciju i u toku izvršavanja. Paralelizacija se može raditi na nivou bazičnih blokova, ali i globalno. Posebno je interesantna paralelizacija programskih petlji, pri čemu je najveći broj radova u literaturi posvećen jednostavnim programskim petljama. Paralelizacija programskih petlji posebno je interesantna jer program najveći deo vremena provodi u izvršavanju programskih petlji. Jedan od najkompleksnijih otvorenih problema je optimizacija programskih petlji sa uslovnim grananjima, a najveće zasluge za modeliranje ove kategorije petlji pripadaju recenzentu ove knjige, profesoru Draganu Milićevu {GSE 89}{MJ 97}{MJ 98}{MJ 00}{MJ 01}{MJ 02}.

Monografija je posvećena paralelizaciji koda na instrukcijskom nivou i obuhvata sve najvažnije oblasti paralelizacije na instrukcijskom nivou, izuzev paralelizacije petlji sa uslovnim grananjima u telu petlji. Pritom je posebno detaljno razrađena paralelizacija komponenti jake povezanosti, jer je autor prvi detaljno opisao taj metod paralelizacije petlji u literaturi {JO 91}.