VPLIV NAČINA ČASOVNE RAZVRSTITVE IZVAJANJA MODULOV ALGORITMA NA POSPEŠITEV PORAZDELJENEGA SISTEMA ZA PROCESNO VODENJE

INFORMATICA 4/92

Vladimir Jovan
Center za tehnologijo vodenja sistemov
Odsek za računalniško avtomatizacijo
in regulacije
Institut Jožef Stefan, Ljubljana

Keywords: industrial process control, distributed control system, scheduling, speedup

Pomembna značilnost računalniških sistemov procesnega vodenja je zahteva po delovanju sistema v realnem času. Načrtovalec lahko temu pogoju zadosti z določitvijo ustrezne arhitekture strojne opreme, v primeru uporabe večračunalniškega sistema pa tudi z izbiro pravilnega zaporedja izvajanja posameznih modulov implementiranega algoritma. V članku opisujemo postopek in rezultate raziskave vpliva načina časovne razvrstitve izvajanja modulov na trajanje cikla algoritma pri porazdeljenem sistemu vodenja poljubnega industrijskega procesa.

INFLUENCE OF TIME SCHEDULING ON SPEEDUP IN A DISTRIBUTED PROCESS CONTROL SYSTEM. The important characteristic of computer process control systems is the capability to run in real-time. This condition can be fulfilled by proper hardware and software design and in case of multicomputer system also by appropriate time scheduling of implemented control algorithm's modules execution. The article describes the procedure and the results of the research which was realized to determine the contribution of proper program module scheduling to the speedup in an industrial distributed control system.

1. UVOD

Naloga računalniškega sistema vodenja industrijskega procesa se je v zadnjih letih bistveno spremenila; iz pasivnega prikazovalca trenutnega stanja vodenega procesa in zapisovalca posameznih dogodkov v procesu se težišče nalog sistema vodenja usmerja predvsem k aktivnemu izvajanju različnih nalog vodenja, samostojnemu odločanju v primeru kritičnih situacij, prilagajanju načina vodenja trenutnim razmeram in doseganju optimalnega režima obratovanja vodenega procesa. V mnogih primerih nekateri razlogi, kakor so narava tehnološkega procesa, fizične razdalje med posameznimi objekti vodenja, zahteve po visoki stopnji zanesljivosti in avtonomnosti delovanja sistema za vodenje, narekujejo uporabo porazdeljenega računalniškega sistema. Zmogljivost enoračunalniškega sistema vodenja je določena z njegovimi tehničnimi značilnostmi (hitrost izvajanja, kapaciteta pomnilnika, itd) in sistem vodenja je operativen le, če implementirani algoritem ne presega časovnih in prostorskih omejitev. Pri porazdeljenem sistemu pa imamo možnost, da prostorske omejitve premostimo z dodajanjem novih procesnih enot v sistem. Časovnim omejitvam zadostimo s sočasnostjo izvajanja modulov algoritma, ki jih porazdelimo med posamezne procesne enote sistema vodenja. Sočasnost izvajanja posameznih modulov algoritma vodenja na porazdeljenem računalniškem sistemu pospeši (angl. *speedup*) izvajanje algoritma, kar daje načrtovalcu sistema vodenja možnost, da z izbiro ustreznega števila procesnih enot zagotovi izvajanje algoritma v realnem času.

Pospešitev izvajanja algoritma vodenja je (ob zanemarjanju tehnoloških značilnosti sistema) odvisna predvsem od števila uporabljenih procesnih enot in od strukture algoritma (od vsebovane vzporednosti).

Znano je, da na stopnjo pospešitve vplivata tudi zaporedje (časovna razvrstitev) izvajanja modulov algoritma in način razmestitve posameznih modulov med procesnimi enotami računalniškega sistema. Stopnjo vpliva razvrstitve na čas izvajanja algoritma kvantitativno definira Grahamov teorem /QUIN87/, ki zagotavlja, da je razmerje časov izvajanja v primeru najslabše oziroma najboljše razvrstitve ob upoštevanju načela kontinuitete izvajanja, manjše od 2 (*načelo kontinuitete izvajanja:* modul, ki je pripravljen za izvajanje, se začne izvajati takoj, ko se v sistemu pojavi prosta procesna enota /EAGE89/). Ker omenjeni teorem določa le zgornjo mejo vpliva razvrstitve in ker imamo z ustrezno razvrstitvijo možnost povečanja učinkovitosti porazdeljenega računalniškega sistema oziroma prihranka na številu uporabljenih procesnih enot v sistemu, smo se

odločili za analizo, ki naj bi podala dejanski vpliv razvrstitve na pospešitev pri porazdeljenih računalniških sistemih za vodenje (industrijskih) procesov.

V ta namen smo programsko opremo, ki predstavlja algoritem vodenja, predstavili z modelom v obliki usmerjenega acikličnega grafa. Na osnovi večjega števila grafov (testne množice) s strukturo, značilno za algoritme vodenja industrijskih procesov, smo določili dejanski vpliv načina razvrstitve na stopnjo pospešitve pri algoritmih za procesno vodenje in si s tem odgovorili na vprašanje o pomenu določitve dobrega načina razvrstitve izvajanja modulov pri zasnovi porazdeljenih računalniških sistemov za vodenje procesov.

2. VLOGA PORAZDELJENIH SISTEMOV PRI VODENJU INDUSTRIJSKIH PROCESOV

Kompleksnost problemov vodenja zahtevnih industrijskih procesov, pojavljanje manjših, cenenih mikroračunalnikov in obstoj zanesljivih komunikacijskih povezav so bili vzroki, ki so omogočili, da se je razvoj računalniško podprtih sistemov za vodenje industrijskih procesov usmeril v uporabo porazdeljenih sistemov. Lahko trdimo, da je prav vodenje procesov eno izmed področij uporabe računalniške tehnologije, kjer se je uporaba distribuiranih sistemov pokazala za najbolj upravičeno in učinkovito.

Lastnosti porazdeljenih sistemov /SLOM87, SCHO84/ omogočajo izpolnjevanje vseh specifičnih zahtev do sistema za vodenje procesa /BENN82/, predvsem v smislu delovanja v realnem času, zanesljivosti obratovanja, dopolnjevanja sistema in vzdrževanja. Namestitev računalnikov neposredno ob samih objektih vodenja operaterjem omogoča avtonomno vodenje in nadzor podsistemov objekta vodenja, zagotavlja hitrejše odzivne čase ter zmanjšuje stroške ožičenja in vpliv motenj. Prav tako je v porazdeljen sistem možna vključitev namenskih procesnih enot, kot tudi povezava z ostalimi računalniškimi sistemi, ki neposredno niso vključeni v proces vodenja proizvodnje.

Osnovna zahteva do sistema vodenja je zmožnost njegovega ***odziva na spremembe stanja vodenega objekta v vnaprej predpisanem času*** (delovanje v realnem času). Ustrezen način odziva določa algoritem vodenja. Ker se algoritem vodenja na posameznem (enoprocesorskem) računalniku lahko izvaja le sekvenčno, lahko pride do prekoračitve časa, znotraj katerega se morajo izvesti vse operacije sistema vodenja kot odgovor na spremembo stanja vodenega objekta in sistem vodenja tako postane s stališča zmožnosti vodenja neuporaben. Vzroka za prekoračitev časovnih omejitev sta predvsem v:

- kompleksnosti funkcij sistema vodenja, ki se morajo izvesti kot odgovor na spremembo stanja v vodenem sistemu,
- veliki dinamiki vodenega sistema, kar se odraža na velikem številu sprememb parametrov sistema v časovni enoti in ustreznem številu potrebnih reakcij sistema vodenja.

Ena izmed rešitev problema je uporaba hitrega (in dragega) računalnika, pri čemer se poleg cene pojavlja še vprašanje primernosti in dobavljivosti take naprave. Drugo rešitev predstavlja razporeditev funkcij med več manj zmogljivih procesnih enot (porazdeljen računalniški sistem), s katerimi skušamo izrabiti možnost sočasnega izvajanja delov algoritma vodenja in s tem zadostiti časovnim omejitvam. Pri načrtovanju sistemov vodenja industrijskih procesov ni poudarek na reševanju problema optimalne arhitekture računalniškega sistema ali problema čimbolj enakomerne zasedenosti posameznih enot, temveč je osnovni cilj doseči pri dani zasnovi strojne opreme in danem algoritmu delovanje sistema vodenja v realnem času, torej izvajanje vseh potrebnih modulov algoritma vodenja znotraj časovnega intervala, ki ga določa narava vodenega procesa.

3. MODEL PORAZDELJENEGA SISTEMA

Splošni model večračunalniškega sistema predstavljata množica procesnih enot in množica modulov razdrobljenega algoritma, ki skupaj zagotavljata izvajanje predpisanih funkcij sistema. Tak model štejemo kot deterministični, če so informacije, ki opisujejo značilnosti elementov sistema, znane vnaprej. Tipične informacije, ki jih potrebujemo, so npr. čas izvajanja posameznega modula, število modulov, število procesnih enot, omejitve v vrstnem redu izvajanja posameznih modulov, lastnosti procesnih enot, itd. Množico resursov (procesnih enot) lahko preprosto definiramo kot množico $P = (P_1, P_2, P_3, ..., P_n)$. Glede na specifične primere so lahko elementi množice P bodisi identični, identični v funkcionalnih zmogljivostih, a različni po hitrosti izvajanja, ali različni tako po funkcijah kot po hitrosti izvajanja.

Množico dekomponiranih modulov splošnega algoritma vodenja lahko v povezavi z množico procesnih enot formalno predstavimo kot peterko $(T, \prec, [\ t_{ij}\ ], R_j, \{\ w_j\ \})$, kjer:

1. $T = (T_1, T_2, ..., T_m)$ predstavlja množico modulov implementiranega algoritma, ki naj se izvaja,
2. " $\prec$ " predstavlja (irefleksivno) relacijo delne urejenosti, ki določa omejitve vrstnega reda izvajanja. Relacija med dvema elementoma T_i in T_j iz množice T, $T_i \prec T_j$ tako določa, da se mora izvajanje modula T_i končati pred začetkom izvajanja modula T_j,
3. $[\ t_{ij}\ ]$ predstavlja matriko časov izvajanja dimenzije n x m, kjer je $t_{ij}>0$ in pomeni čas, potreben za izvedbo modula T_j, $1 \le j \le m$ na procesni enoti P_i, $1 \le i \le n$. Če velja $t_{ij} = \infty$, to

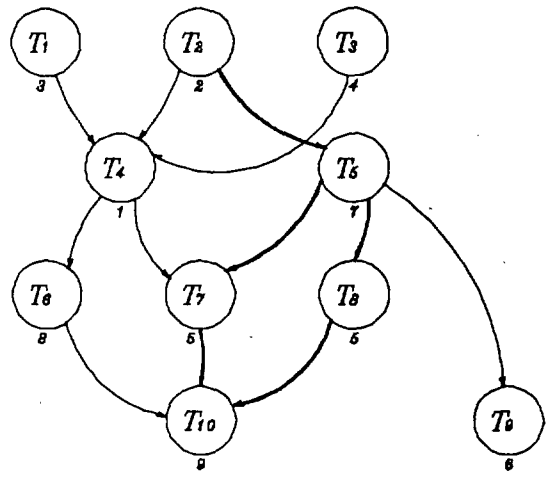
pomeni, da se modul T_j ne more izvajati na procesni enoti P_i. Za vsak j obstoja vsaj en i tak, da $t_{ij} < \infty$. V primeru identičnosti procesnih enot nam t_j predstavlja čas izvajanja modula T_j na eni od procesnih enot. Začetni čas izvajanja modula označimo z s_{ij} in čas, ko se konča izvajanje modula T_j na procesni enoti P_i z f_{ij}.

4. $R_j = [\ R_1(T_j), \ldots, R_s(T_j)\ ]$, $1 \le j \le m$, določa količino dodatnih resursov R, potrebnih za izvedbo modula T_j.
5. uteži $\{\ w_i\ \}$, $1 \le i \le m$, so poljubne funkcije razvrstitve Ψ in predstavljajo ceno, ki jo določa specifična razvrstitev Ψ glede na posamezen modul T_i. Navadno jih tretiramo kot konstante. Tako npr. velja, da je "cena" izvajanja modula T_i v času t preprosto $w_i.t$.

Opisana formulacija sistema modulov splošnega algoritma vsebuje precej več posplošitev, kot jih nameravamo uporabljati v naši analizi. Večina avtorjev uporablja za določitev splošnega algoritma le točke 1. do 3., to je število modulov, njihovo medsebojno odvisnost v smislu časovnega zaporedja izvajanja in čas, potreben za izvajanje posameznega modula.

Relacijo delne urejenosti $\prec$ je zelo primerno predstaviti v obliki ***usmerjenega acikličnega grafa***, kjer jo predstavimo z množico povezav med vozlišči grafa /TENE86, McHU90, QUIN87/. Z dekompozicijo algoritma na posamezne module razbijemo enovit algoritem v množico modulov $T=(T_1, T_2, \ldots, T_m)$. Če obstojajo medsebojne povezave med moduli (podatkovni pretoki, sinhronizacijski signali, itd), potem eksistira neka delna urejenost med moduli, ki jo definira binarna relacija imenovana "***precedenčnost***" (prevod: *predhodnost, prednost (v času)*), ki delno definira potrebni vrstni red izvajanja posameznih modulov. Množico modulov algoritma in precedenčne relacije med njimi lahko ponazorimo z grafom $G(V,A)$ (slika 1), kjer V določa množico uteženih vozlišč grafa (modulov algoritma) in A množico usmerjenih povezav v grafu (podatkovne, sinhronizacijske in druge povezave), ki ponazarjajo precedenčne relacije.

V ***usmerjenem uteženem acikličnem grafu*** lahko določimo najdaljšo pot iz začetnega vozlišča do končnega vozlišča in na ta način izračunamo minimalni možni čas, ki nam v splošnem zagotavlja izvršitev (enega cikla) algoritma. Minimalni možni čas algoritma je enak kritični poti v grafu /CONW67, McHU90/ in ni odvisen od števila uporabljenih računalnikov v sistemu, temveč samo od časa izvajanja posameznih modulov in njihovih medsebojnih povezav. Na osnovi analize kritične poti lahko analitično ugotovimo, če je v algoritmu zadostna stopnja paralelizma, ki pri določeni strojni opremi omogoča izvajanje algoritma v predpisanem času.



Slika 1: *Primer usmerjenega uteženega acikličnega grafa*

4. RAZVRSTITEV IZVAJANJA MODULOV

Časovno zaporedje izvajanja posameznih modulov je določeno s tem, da se posamezen modul lahko začne izvajati takrat, ko so se izvedli vsi njegovi predhodniki in ko obstoja prosta procesna enota, na kateri se modul lahko izvaja. Vrstni red izvajanja v splošnem ni enolično določen, saj lahko hkrati nastopa več modulov, ki nimajo (neizvedenih) predhodnikov in so tako kandidati za izvajanje. Potrebna je neka strategija ***določitve vrstnega reda izvajanja*** Ψ_T in ***razmestitve*** Ψ_P izvajanja modulov po posameznih procesnih enotah, ki omogoča pravilno delovanje algoritma ob upoštevanju danih omejitev in jo lahko imenujemo ***strategija razvrstitve*** Ψ. Problem razvrstitve izvajanja modulov algoritma v večračunalniškem sistemu vsebuje torej:

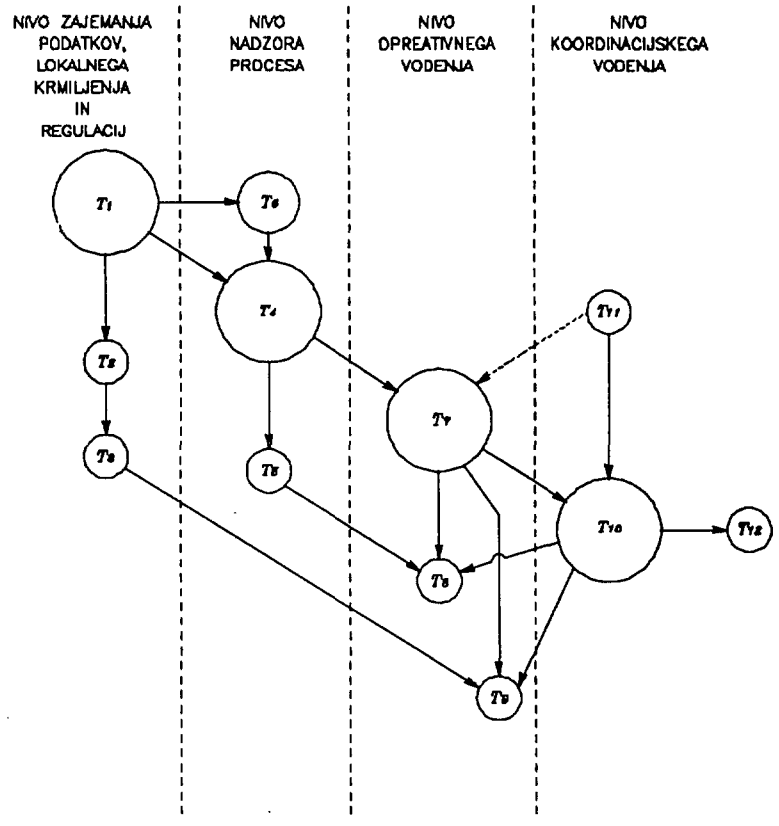
- časovno komponento problema, ki določa časovno razvrstitev izvajanja modulov,
- prostorsko komponento problema, ki rešuje vprašanje mesta izvajanja posameznega modula v celotnem sistemu,

torej $\Psi = f(\Psi_T, \Psi_P)$.

Pri načrtovanju sistemov vodenja industrijskih procesov so sistemske funkcije znane vnaprej; torej so znane tudi značilnosti programskih modulov kot preslikave sistemskih funkcij v realizirano programsko opremo. Zato razvrstitev modulov algoritma sistema procesnega vodenja po eni izmed uveljavljenih klasifikacij /CASA88/ opredelimo kot statično, globalno in suboptimalno, ker jo ob poznanih značilnostih modulov (čas trajanja, načini in vrste povezav) lahko za vse module izvedemo vnaprej, za rezultat razvrstitve pa ni nujno, da je optimalen, temveč je lahko suboptimalen v smislu zagotavljanja izvajanja algoritma znotraj predpisanih časovnih omejitev.

5. VPLIV NAČINA RAZVRSTITVE ALGORITMA VODENJA NA STOPNJO POSPEŠITVE

Pri računalniško vodenem procesu je računalniški sistem objekt, ki obvladuje informacijski tok in ga v skladu z zahtevami vodenja zbira, prilagaja, usmerja, transformira in shranjuje. Način izvajanja teh funkcij določa algoritem vodenja kot sestavni del računalniškega sistema vodenja. Algoritem vodenja je za vsak konkreten industrijski proces specifičen, vendar ima skupne osnovne značilnosti, ki izhajajo iz koncepta vodenja procesov oziroma nalog, ki jih izvaja sistem vodenja /MATK87/. Za izvedbo vodenja mora algoritem vodenja omogočati sprotno zajemanje procesnih podatkov, njihovo obdelavo, prikaz, izvedbo različnih računskih in logičnih operacij nad podatki z namenom delovanja krmilnih, regulacijskih, nadzornih, optimizacijskih in drugih funkcij sistema, pošiljanje krmilnih in regulacijskih signalov v vodeni proces, kratkoročno in dolgoročno shranjevanje podatkov in povezavo sistema vodenja z okoljem. Posamezne funkcije splošnega algoritma vodenja so realizirane s programskimi moduli, ki v iteraciji izvajajo operacije nad podatki in katerih začetek izvajanja je pogojen z dostopnostjo potrebnih vhodnih podatkov, ki so bodisi merjene procesne veličine, ročno vpisani podatki ali ukazi, rezultati, ki jih posredujejo drugi moduli, itd. Če module tretiramo kot vozlišča in podatkovne povezave med moduli opišemo z usmerjenimi povezavami med vozlišči, lahko splošno strukturo algoritma vodenja prikažemo z usmerjenim acikličnim grafom (slika 2), iz katerega je razvidna tudi hierarhična soodvisnost funkcij vodenja.



T1 – zajemanje podatkov iz procesa,predobdelava podatkov
T2 – enostavni algoritmi lokalnega krmiljenja in regulacije
T3 – pošiljanje izhodnih signalov lokalnega krmiljenja in regulacije v proces
T4 – ugotavljanje trenutnega stanja procesa
T5 – obravnavanje alarmov
T6 – prikaz trenutnega stanja procesa
T7 – operativno vodenje procesa,lokalna optimizacija
T8 – shranjevanje podatkov,protokoliranje
T9 – pošiljanje izhodnih signalov operativnega vodenja v proces
T10 – koordinacija,globalna optimizacija
T11 – sprejem podatkov iz drugega (višjega) sistema vodenja
T12 – oddaja podatkov drugemu (višjemu) sistemu vodenja

Slika 2: *Struktura splošnega algoritma vodenja v obliki splošnega usmerjenega acikličnega grafa*

Začetna vozlišča grafa algoritma vodenja predstavljajo moduli za zajemanje procesnih podatkov in sprejemanje podatkov oziroma navodil z drugega hierarhično višjega sistema vodenja. Končna vozlišča grafa so moduli za pošiljanje izhodnih signalov v proces, shranjevanje podatkov o stanju sistema, protokoliranje in povezavo z hierarhično višjim nivojem vodenja. Trajanje enega cikla je odvisno od konkretne izvedbe algoritma vodenja, v splošnem pa lahko rečemo, da kritična pot vodi od zajemanja procesnih podatkov, ugotavljanja trenutnega stanja sistema, operativnega vodenja do koordinacije vodenja procesa z okolico.

Obsežnost, vsebina in čas izvajanja posameznega modula so odvisni od konkretne realizacije vodenja na določenem procesu. V splošnem velja, da moduli na nižjem nivoju izvajajo vrsto enostavnejših in medsebojno neodvisnih operacij nad velikim številom potrebnih podatkov, moduli na višjih nivojih pa kompleksnejše algoritme nad relativno manjšim naborom podatkov. Zato je možna razdrobitev osnovnih modulov algoritma vodenja na več manjših neodvisnih ali šibko podatkovno povezanih modulov in njihova razvrstitev med več procesnih enot, kar pa spet nudi možnosti sočasnosti izvajanja določenih modulov in s tem povezanega hitrejšega izvajanja algoritma vodenja.

Iz teorema o razmerjih časov izvajanja pri različnih razvrstitvah /ONDA84/ lahko ugotovimo, da poljuben način razvrstitve ob upoštevanju načela kontinuitete izvajanja zagotavlja čas izvajanja cikla algoritma, ki je

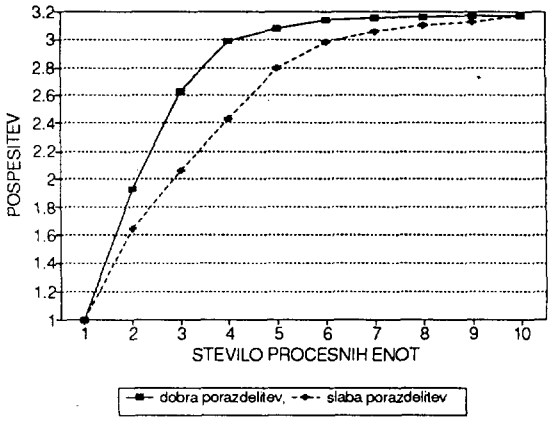
lahko do dvakrat daljši od časa izvajanja pri optimalni razvrstitvi. Zato je način razvrstitve pomemben dejavnik pri načrtovanju sistemov vodenja, posebno v mejnih primerih, ko primerna razvrstitev lahko odločilno skrajša čas izvajanja ali pa prihrani vgradnjo dodatne procesne enote v sistem. Prav tako primerna razvrstitev omogoča zmanjšanje pretoka podatkov med procesnimi enotami, kar razbremeni tako same procesne enote, predvsem pa zasedenost komunikacijske mreže.

V redkih primerih lahko dosežemo, da smiselna dekompozicija algoritma vodenja da kot rezultat množico modulov, ki so po času izvajanja enaki. Še redkeje so precedenčne relacije med dekomponiranimi moduli algoritma take, da relacija delne urejenosti predstavlja drevo. V obeh primerih imamo na voljo algoritma, ki dajeta optimalen rezultat /ONDA84/. V vseh ostalih različnih načinih dekompozicije algoritma, ki se odražajo na stopnji izkoriščenosti možnega paralelizma, velikosti posameznih modulov in različnih oblikah precedenčnih povezav, pa nimamo na voljo pravil, ki bi vnaprej zagotavljala uspešnost določenega načina razvrstitve. Nekateri, sicer uspešni postopki razvrstitve, včasih privedejo do vrste anomalij /COFF76/. Zato lahko uspešnost pristopa k problemu razvrstitve po določeni metodi lahko ocenimo le na osnovi uporabe metode na večjem številu primerov.

Ker je velika večina problemov razvrstitve NP-polnih /COFF76/, ni presenetljivo, da se teorija razvrstitve v zadnjem času usmerja na iskanje učinkovitih ***hevrističnih postopkov***. Ena zelo popularnih hevrističnih metod razvrstitve modulov je metoda imenovana LPT (angl. *Largest Processing Time*) ali HNF (angl. *Heavy Nodes First*), ki temelji na prednostnem izvajanju tistih modulov, ki so s stališča časa izvajanja največji /CONW67, SHIR89/. Metodo razvrščanja LPT smo uporabili tudi v našem primeru. Postopek časovnega razvrščanja po metodi LPT smo dopolnili z upoštevanjem nivoja vozlišča v grafu, ki v primeru, da imamo med moduli več kandidatov za izvajanje z enakimi časi, prednostno izbere za izvajanje modul z najvišjim nivojem, to je z največjo oddaljenostjo vozlišča od končnega vozlišča. Na testni množici 50-ih različnih grafov s strukturo, značilno za algoritme procesnega vodenja in s po 30-imi vozlišči, katerih uteži so bile naključno izbrane v velikostnem redu od 1 do 9, smo razvrstitev izvedli na dva načina:

- prvič po metodi LPT z upoštevanjem nivojev posameznih vozlišč,
- drugič po postopku, ki daje prednost izvajanju modulom z najmanjšim časom izvajanja in najnižjim nivojem.

Pri obeh metodah smo upoštevali načelo kontinuitete izvajanja. Na ta način smo določili za vse grafe v celotni testni množici stopnjo pospešitve v odvisnosti od načina razvrstitve zaporedja izvajanja posameznih modulov. Ker je znano, da je metoda LPT ena izmed hevrističnih metod razvrščanja, ki daje dobre rezultate glede na čas trajanja algoritma, in ker smo eksperimentalno ugotovili, da razvrščanje po drugi metodi daje slabe rezultate glede na ostale hevristične metode, smo na ta način dobili razmerje med povprečnimi časi izvajanja algoritma v primeru ***slabega*** in ***dobrega*** načina razvrščanja, ki ga navzgor omejuje že opisani Grahamov teorem na vrednost 2. Slika 3 prikazuje razliko med povprečjem doseženih vrednosti pospešitve v primeru uporabe dobre oziroma slabe strategije ravrstitve.



Slika 3: *Pospešitev v primeru dobre (HNF) in slabe porazdelitve pri številu modulov m=30 in različnem številu procesnih enot*

Povprečna razlika med doseženimi vrednostmi pospešitve je pri določenem številu uporabljenih procesnih enot več kot 20 odstotkov. Razlika stopnje pospešitve kot posledica dobrega in slabega načina razvrstitve je prisotna pri vseh grafih testne množice (npr. pri številu razpoložljivih procesnih enot n=2 je bila najmanjša razlika obeh pospešitev 7.5 odstotkov in največja razlika 30.3 odstotka). To v praksi potrjuje vpliv uporabljenega načina razvrstitve na pospešitev, po drugi strani pa potrjuje dejstvo, da vpliv porazdelitve ni tako velik, kot bi ga lahko pričakovali na osnovi Grahamovega teorema.

6. ZAKLJUČEK

Osnovna zahteva, ki jo mora načrtovalec računalniškega sistema procesnega vodenja v fazi načrtovanja upoštevati, je zmožnost delovanja sistema v realnem času. Predpisanim časovnim (in prostorskim) omejitvam lahko zadostimo z uporabo porazdeljenega računalniškega sistema, kar pa nam daje dodatno možnost, da z ustrezno razvrstitvijo vrstnega reda izvajanja posameznih modulov algoritma vodenja še zmanjšamo čas izvajanja (enega cikla) celotnega algoritma vodenja.

Namen pričujočega članka je bil praktično ovrednotiti vpliv razvrstitve časovnega izvajanja modulov na pospešitev izvajanja algoritma na porazdeljenem

računalniškem sistemu. Čeprav je stopnja pospešitve odvisna predvsem od števila uporabljenih procesnih enot, strukture samega algoritma vodenja in količine podatkov, ki se prenašajo med posameznimi enotami porazdeljenega sistema, smo ugotovili, da vsaj pri algoritmih s strukturo, ki je značilna za algoritme procesnega vodenja, vpliv načina razvrstitve izvajanja modulov na pospešitev ni zanemarljiv. Zato je v fazi načtovanja porazdeljenega sistema procesnega vodenja za konkreten algoritem vodenja potrebno določiti ustrezen vrstni red izvajanja posameznih modulov. S skrajšanjem časa izvajanja cikla algoritma dosežemo zanesljivejše delovanje sistema, v določenih primerih pa z ustrezno razvrstitvijo izvajanja modulov lahko zmanjšamo število uporabljenih procesnih enot v sistemu.

7. LITERATURA

/BENN82/: S. Bennet, D. A. Linkens: ***Computer Control of Industrial Processes,*** Peter Peregrinus Ltd, 1982

/CASA88/: T. L. Casavant, J. G. Kuhl: ***A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems***, IEEE Transactions on Software Engineering, Vol. 14, No. 2, 1988

/CONW67/: R. W. Conway, W. L. Maxwell, L. W. Miller: ***Theory of Scheduling***, Addison-Wesley Inc., 1967, pp. 132 - 140

/COFF76/: E. G. Coffman: ***Computer and Job-Shop Scheduling Theory***, John Wiley and sons, Inc., New York, 1976, pp. 2 - 50

/EAGE89/: D. L. Eager, J. Zahorjan, E. D. Lazowska: ***Speedup Versus Efficiency in Parallel Systems***, IEEE Transactions on Computers, Vol. 38, No. 3, marec 1989

/MATK87/: D. Matko, B. Zupančič: ***Računalniški sistemi v vodenju procesov***, 3. izdaja, Univerza v Ljubljani, Fakulteta za elektrotehniko, 1987

/McHU90/: J. A. McHugh: ***Algorithmic Graph Theory***, Prentice Hall Inc., 1990

/ONDA84/: J. Ondaš: ***Algorithms for Scheduling Homogeneus Multiprocessor Computers***, Algorithms,Software and Hardware of parallel Computers, uredila J. Mikloško in V. E. Kotov, VEDA, 1984

/QUIN87/: M. J. Quinn: ***Designing Efficient Algorithms for Parallel Computers***, McGraw-Hill Inc, 1987

/SCHO84/: J. D. Schoeffler: ***Distributed Computer Systems for Industrial Process Control***, IEEE Computer, February 1984

/SHIR89/: B. Shirazi, M. Wang: ***Heuristic Functions for Static Task Allocation***, Microprocessing and Microprogramming 26, 1989

/SLOM87/: M. Sloman, J. Kramer: ***Distributed Systems and Computer Networks***, Prentice-Hall Int., 1987

/TENE86/: A. M. Tenenbaum, M. J. Augenstein: ***Data Structures Using Pascal***, Prentice-Hall Inc., 1986, pp. 575-643