STOHASTIČNI PRISTOP K ANALIZI UČINKOVITOSTI VEČPROCESORSKIH SISTEMOV

INFORMATICA 1/88

Slavko Mavrič
Peter Kolbezen
Institut »Jožef Stefan«

UDK 681.324:519.21

Predstavljen je stohastični pristop pri modeliranju in analizi učinkovitosti večprocesorskih sistemov s skupnim pomnilnikom, pri katerih je vsakemu procesorju pridružen še njegov zasebni pomnilnik. Zgrajen je model večprocesorskega sistema in definiran njegov režim delovanja. Postopek analize temelji na opisu tega modela s stohastično Petrijevo mrežo. Iz drevesa dosegljivosti te mreže dobimo parametre pripadajoče markovske verige. Analiza stacionarnega stanja te verige pa nas pripelje do iskanega pokazatelja učinkovitosti večprocesorskega sistema.

Stochastic Approach in Performance Analysis of Multiprocessor Systems. A stochastic approach in modelling and performance analysis of global memory based multiprocessor systems is presented. Every processor of such a system possesses its own private memory. A model of system has been built and its workload has been defined. Description of system activity with stochastic Petri net is the first step in a performance analysis. From the reachability tree of this Petri net it is possible to obtain parameters of associated Markov chain. Steady state analysis of this chain leads us finally to a desired performance index of a multiprocessor system.

1. Uvod

Pri ocenjevanju učinkovitosti računalniških sistemov srečamo tri osnovne pristope. To so: merjenje, simulacija in analitično modeliranje.

Prvi pristop vključuje meritve na samem sistemu pod dejanskimi delovnimi pogoji ter primerjalno merjenje s testnimi (benchmark) programi. Oboje zahteva popolno okolje ocenjevanega sistema.

Pri simulaciji opišemo dogajanje v računalniškem sistemu s simulacijskim programom. Pri tem so nam navadno v pomoč temu namenjeni programski jeziki. Iz izvajanja simulacijskega programa sklepamo o lastnostih simuliranega sistema.

Analitični model predstavlja z matematičnimi orodji opisano dogajanje v sistemu. Ocena učinkovitosti sistema nastopa kot analitična ali numerična rešitev tega modela. Velik pomen analitičnega modeliranja je v dejstvu, da ga lahko opravimo že v zelo zgodnji fazi načrtovanja računalniškega sistema in nam je v pomoč pri izbiri optimalne arhitekture.

V tem članku predstavljamo stohastični pristop k modeliranju in analizi večprocesorskih sistemov na osnovi markovskih procesov.

2. Določanje modela večprocesorskega sistema

Prvi korak pri ocenjevanju učinkovitosti nekega večprocesorskega sistema je v določitvi njegovega modela, ki mora zajeti vse bistvene lastnosti sistema. Model mora definirati množico sistemskih gradnikov, njih medsebojne povezave in razmerja ter režim delovanja celotnega sistema. Pri tem se moramo odločiti za primeren nivo abstrakcije predstavitve sistema. Nivo abstrakcije pomeni nivo detajlov, ki jih pri opisu sistema upoštevamo. Ta naj določa kaj so gradniki sistema ter kakšna so njihova medsebojna razmerja in pravila komunikacije. Nivo abstrakcije izberemo tako, da nam bo analiza na osnovi izbranega modela dala kot rezultat nek parameter, ki bo dovolj dobro opisoval učinkovitost sistema. Model mora torej vsebovati vse bistvene elemente, ki so za analizo potrebni, vse detajle, ki na tem nivoju abstrakcije niso pomembni, pa naj model ne zajame. Pri vrednotenju učinkovitosti računalniških sistemov lahko zasledimo različne nivoje abstrakcije:

- Aparaturni nivo. Ta nivo se nanaša na preizkušanje pravilosti nekega vezja z logičnega stališča. Gradniki modela tega nivoja so posamezna integrirana vezja. Za opis takega modela obstajajo posebni programski jeziki.
- Funkcionalni nivo. Cilj analize na tem nivoju je ovrednotenje obnašanja osnovnih funkcionalnih enot aparaturne opreme, kot so procesorji, pomnilniške enote, vodila itd., ko ti sodelujejo pri izvajanju neke operacije. Opis modela je podan s parametri, ki govore o hitrosti, kapaciteti in zakasnitvi posameznih enot.
- Sistemski nivo. Na tem nivoju se ovrednoti učinkovitost celotnega sistema na osnovi poznanih funkcij posameznih enot. Osnovni gradniki modela so aparaturno/programske enote, kot procesne enote, vhodno/izhodne in komunikacijske enote.

Izbira najprimernejšega nivoja abstrakcije pri modeliranju nekega sistema je odvisna od razmerja med točnostjo rezultatov na eni strani in kompleksnostjo ter ceno analize na drugi strani. Za ovrednotenje učinkovitosti večprocesorskega sistema nam ni potrebno poznati detajlov na nivoju posameznih integriranih vezij, prav tako pa za splošno namenski sistem ne potrebujemo natančno poznavanje aktivnosti sistema v realnem okolju neke aplikacije. Zato je funkcionalni nivo za analizo učinkovitosti večprocesorskih sistemov najprimernejši.

Osnovne predpostavke, na katerih temelji naš model večprocesorskega sistema so naslednje:

- osnovni gradniki modela so procesorji, pomnilniške enote in povezovalna mreža.

- vsak procesor P ima svoj zasebni pomnilnik ZP, ki je dostopen le njemu.
- vsak procesor lahko zahteva dostop do neke skupne pomnilniške enote SP, ki mu je dostopna.
- pomnilniška enota sprejme in po njej lastnem pravilu servisira zahteve posameznih procesorjev.
- povezovalna mreža PM povezuje procesorje in skupne pomnilniške enote. Če ni drugače rečeno, se predpostavlja, da so zakasnitve, ki jih vnaša povezovalna mreža, enake nič.

Splošen model, ki zadošča tem pogojem prikazuje slika 1.

Ko je nivo abstrakcije določen in so s tem poznani gradniki modela in njih medsebojne povezave, je potrebno definirati še režim delovanja tega modela. Pri sistemih, v katerih je vsakemu procesorju pridružen zasebni pomnilnik, izvajajo procesorji program iz tega pomnilnika. To izvajanja se prekinja s posegi v skupni pomnilnik. Aktivnost posameznega procesorja lahko smatramo kot zaporedje dostopov do različnih pomnilniških enot. Model režima delovanja nekega procesorja je torej v ponavljajočem izvajanju naslednje sekvence:
- dostop, ali zaporedje dostopov v zasebni pomnilnik
- dostop, ali zaporedje dostopov v skupni pomnilnik

Konfliktna situacija lahko nastopi pri uporabi povezovalne mreže ali skupnih pomnilniških enot. Vsak procesor je tekom delovanja v splošnem v enem izmed naslednjih treh stanj:
- Aktivno stanje. Procesor izvaja program iz svojega zasebnega pomnilnika.
- Stanje dostopa. Procesor izvaja bralno ali vpisovalno operacijo na neki skupni pomnilniški enoti.
- Čakajoče stanje. Procesor čaka na zahtevani dostop do skupne pomnilniške enote.

Takšna aktivnost procesorjev na funkcionalnem nivoju je lahko posledica povsem različnega obnašanja na sistemskem nivoju.

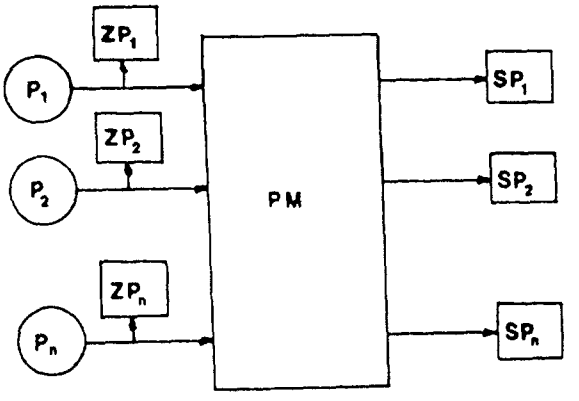
Zamislimo si tri večprocesorske sisteme. Pri prvem poteka komunikacija med posli s pošiljanjem sporočil preko pomnilniških vmesnikov, lociranih v skupnem pomnilniku. Pri drugem sistemu izvajajo procesorji neko operacijo nad vhodnimi podatki, ki jih najdejo v skupnem pomnilniku; tja tudi shranjujejo vmesne rezultate. V tretjem sistemu pa izvajajo procesorji program iz hitrih 'cashe' pomnilnikov, ki se morajo občasno osveževati iz skupnega pomnilnika. Če opazujemo obnašanje posameznega procesorja v vseh treh sistemih, zaključimo, da v vsakem primeru le-to rezultira v zaporedju dostopov do različnih pomnilniških enot. Zaporedju dostopov v zasebni pomnilnik sledi zaporedje dostopov v skupni pomnilnik in obratno.

Ti trije primeri kažejo, kako se lahko povsem različno delovanje večprocesorskega sistema na sistemskem nivoju, reducira na enako obnašanje na funkcionalnem nivoju. Režim delovanja na funkcionalnem nivoju je torej v vseh treh primerih enak.

Za ocenitev učinkovitosti večprocesorskega sistema na osnovi zgrajenega modela, moramo seveda najprej količinsko oceniti parametre tega modela. Pri našem funkcionalnem modelu so ti parametri časi trajanja zaporedij dostopov do zasebnega in skupnega pomnilnika, ki v celoti popisujejo režim delovanja posameznih procesorjev. Tej oceni sledi postopek analize. Ta nas na osnovi teh parametrov in topološkega opisa sistema vodi k izračunu veličine, ki nam bo dala dovolj dober opis učinkovitosti večprocesorskega sistema. Imenujemo jo performančni pokazatelj.

Pri določitvi režima delovanja, oziroma pri ocenitvi parametrov modela srečamo dva osnovna pristopa.

Deterministični pristop zahteva natančno poznavanje delovanja vsakega procesorja. Dobra stran tega pristopa je v tem, da v popolnosti



Slika 1. Splošen model večprocesorskega sistema

sledimo obnašanju celega sistema in s simulacijo določamo čas, ki je potreben za dokončanje neke poznane operacije. Slabost determinističnega pristopa je v dejstvu, da je potrebno popolno poznavanje režima delovanja procesorjev, kar v času načrtovanja sistema ni vedno možno in da so dobljeni rezultati relevantni le za tisto aplikacijo in ne kažejo funkcijske odvisnosti od parametrov modela.

Če uporabimo verjetnostni pristop, obravnavamo parametre režima delovanja kot naključne spremenljivke z neko znano porazdelitvijo verjetnosti. Stohastični pristop nam ne omogoča tako natančnega vpogleda v izvajanje neke operacije v sistemu kot deterministični pristop, zato pa nam daje rezultate kot funkcijske odvisnosti med parametri modela in performančnimi pokazatelji. S tem nam pomaga k boljšemu razumevanju dogajanja v kompleksnem večprocesorskem sistemu. Stohastično analizo je moč opraviti že v zelo zgodnji fazi razvoja samega sistema kot pomoč pri testiranju in primerjanju posameznih arhitekturnih različic s stališča izbranih performančnih pokazateljev. Verjetnostni pristop temelji na tem, da opišemo dogajanje v modelu z nekim stohastičnim procesom. Število stanj tega procesa je odvisno od števila vseh možnih stanj, ki jih lahko doseže obravnavani večprocesorski sistem, oziroma njegov model. Določiti moramo tudi medsebojne odvisnosti, oziroma pogojne verjetnosti med stanji procesa. Zaradi sprejemljive matematične kompleksnosti se za modeliranje podobnih sistemov pogosto uporablja razred stohastičnih procesov z markovsko lastnostjo, ki jim pravimo markovske verige.

3. Markovske verige

Definicije in rezultati tega poglavja se nanašajo na tip časovno zveznih markovskih verig (ČZMV). Pri njih lahko prihaja do prehodov med stanji v vsakem trenutku, medtem ko so prehodi pri časovno diskretnih markovskih verigah možni le ob posameznih trenutkih, ki so določeni z diskretizacijo časovne osi. Zato so zvezne verige primernejše za opisovanje dogajanja v nekem večprocesorskem sistemu.

Definicijo ČZMV, ki opisuje omenjeno markovsko lastnost, lahko zapišemo takole. Stohastični proces $\{X(t),\ t\geq 0\}$ je ČZMV če velja

$$P\{X(t_{n+1})=x_{n+1} \mid X(t_n)=x_n, \ldots, X(t_0)=x_0\} =$$

$$= P\{X(t_{n+1})=x_{n+1} \mid X(t_n)=x_n\},$$

$$t_{n+1}>t_n>t_{n-1}>\ldots>t_0. \qquad (3.1)$$

Ta markovska lastnost pomeni, da je verjetnostno

obnašanje procesa v prihodnosti odvisno samo od trenutnega stanja in ne tudi od zgodovine procesa. Desna stran gornje enačbe predstavlja prehodno verjetnost med dvema stanjema verige, ki jo za primer homogene ČZMV (kjer so prehodne verjetnosti neodvisne od časa opazovanja) lahko zapišemo v obliki

$$p_{ij}(\tau) = P\{X(t+\tau)=j \mid X(t)=i\}. \qquad (3.2)$$

Vse prehodne verjetnosti verige lahko zapišemo v obliki matrike $P(t) = [p_{ij}(t_1)]$. Seveda velja enakost

$$\Sigma_j\, p_{ij}(\tau) = 1, \quad j \in S. \qquad (3.3)$$

Verjetnost stanja i v trenutku t je po teoremu o popolni verjetnosti podana z enačbo

$$\pi_i(t) = \Sigma_j(p_{ji}(t)\pi_j(0)). \qquad (3.4)$$

iz katere vidimo, da je obnašanje procesa popolnoma določeno z začetno porazdelitvijo in prehodnimi verjetnostmi.

Naj bo

$$r_{ji}(\tau) = p_{ji}(\tau) \qquad j <> i \qquad \text{in}$$

$$r_{ji}(\tau) = p_{ji}(\tau)-1 \qquad j = i. \qquad (3.5)$$

Dokažemo lahko, da obstaja pri ČZMV s končnim številom stanj limita

$$q_{ij} = \lim_{\tau->0} (r_{ij}(\tau)/\tau). \qquad (3.6)$$

q_{ij} predstavlja v primeru i <> j časovno gostoto prehodov s katero proces prehaja iz stanja i v stanje j, medtem ko pomeni $-q_{ii}$ v primeru i = j časovno gostoto s katero proces zapušča stanje i. Velja enakost:

$$\Sigma_j\, q_{ij} = 0, \quad j \in S. \qquad (3.7)$$

Iz enačbe 3.4 lahko, s tako definiranimi qij, pridemo do enačbe za verjetnost nekega stanja i v poljubnem trenutku.

$$\pi'_i(t) = \Sigma_j(q_{ji}\pi_j(t)), \quad j \in S. \qquad (3.8)$$

Dobili smo torej sistem diferencialnih enačb, katerega rešitev nas vodi k verjetnosti posameznih stanj v poljubnem trenutku t. Zaradi težavnosti analitičnega reševanja takega sistema, uporabimo navadno numerične integracijske metode.

Kadar pri neki ČZMV obstaja stacionarna porazdelitev verjetnosti, nas ta navadno bolj zanima in je tudi lažje izračunljiva kot porazdelitev v poljubnem času t. Izkaže se, da obstaja stacionarna porazdelitev pri neskrčljivih homogenih ČZMV in da je neodvisna od začetne porazdelitve. Stacionarna verjetnost nekega stanja je enaka limitni vrednosti njegove verjetnosti, torej

$$\pi_i = \lim_{t->\infty} \pi_i(t). \qquad (3.9)$$

Tedaj tudi velja

$$\lim_{t->\infty} \pi'_i(t) = 0 \qquad (3.10)$$

in tako pridemo do sistema linearnih enačb

$$\Sigma_j(q_{ji}\pi_j) = 0, \quad j \in S, \qquad (3.11)$$

Ker je ta sistem enačb homogen, obstaja rešitev $\pi_i = 0$ za vsak $i \in S$. Če so enačbe linearno neodvisne, je to tudi edina rešitev sistema in stacionarna porazdelitev ne obstaja. Če pa so enačbe linearno odvisne, dobimo stacionarno porazdelitev z dodatkom normalizacijskega pogoja

$$\Sigma_i\, \pi_i = 1, \quad i \in S. \qquad (3.12)$$

Prav analiza stacionarnega stanja, oziroma izračun stacionarnih verjetnosti stanj časovno zvezne markovske verige je velikega pomena pri ovrednotenju performanc večprocesorskih sistemov. Zato bomo rezultate te analize koristno uporabili v nadaljnjem izvajanju.

Poglejmo si še, po kakšnem zakonu so porazdeljeni časi trajanja posameznih stanj markovske verige. Markovska lastnost pravi, da je obnašanje procesa v prihodnosti odvisno le od stanja v katerem se proces nahaja v trenutku opazovanja t, ne pa tudi od zgodovine procesa. Prihodnost procesa torej tudi ni odvisna od časa, ki ga je proces prebil v tem stanju pred časom t. Za naključno spremenljivko W_i, ki pomeni čas trajanja stanja i torej velja naslednja lastnost

$$P\{W_i>t+\tau \mid W_i>t\} = P\{W_i>\tau\}. \qquad (3.13)$$

Edini porazdelitveni zakon, ki zadošča temu pogoju, je eksponentna porazdelitev. Gostoto verjetnosti te porazdelitve podaja enačba

$$f(t) = a \exp(-at), \qquad t >= 0. \qquad (3.14)$$

Lahko bi pokazali, da so časi trajanja posameznih stanj ČZMV eksponentno porazdeljeni s funkcijo gostote verjetnosti

$$f_{Wi}(t) = -q_{ii} \exp(q_{ii}t), \qquad t>=0 \qquad (3.15)$$

in s srednjo vrednostjo

$$E(W_i) = -1/q_{ii}. \qquad (3.16)$$

Analiza večprocesorskih sistemov s pomočjo pridruženih markovskih verig pomeni torej predpostavko, da se čase trajanja aktivnih stanj in stanj dostopa obravnava kot eksponentno porazdeljene naključne spremenljivke.

4. Stohastične Petrijeve mreže

V nadaljevanju nas zanima vprašanje, kako na osnovi modela večprocesorskega sistema pridemo do pridružene markovske verige. Direktna pot je lahko zelo zapletena, saj moramo evidentirati vsa stanja sistema in časovne gostote prehodov med njimi. Pomagamo si s stohastičnimi Petrijevimi mrežami (SPM), ki predstavljajo učinkovito orodje za opisovanje sočasnosti in sinhronizacije v paralelnih sistemih. SPM so nekakšna nadgradnja standardnih Petrijevih mrež v tem, da vpeljejo vanje dimenzijo časa kot naključno veličino.

SPM je bipartiten graf, ki ga sestavlja množica položajev P, množica prehodov T, množica povezav A, množica prehodnih hitrosti G in začetna označitev M_0. SPM je torej določena s peterko (P, T, A, G, M_0). Vsak element iz množice G je prirejen enemu elementu iz množice T, kar pomeni, da je za vsak prehod podana njegova prehodna hitrost. Če je ta večja od nič, govorimo o časovnem prehodu, če pa je enaka nič, pa o takojšnjem prehodu. V grafični ponazoritvi rišemo položaje kot kroge, časovne prehode kot mastne črte, takojšnje prehode kot tanke črte, povezave pa kot puščice. Povezave vežejo položaje s prehodi in prehode s položaji. Položaj p_i je vhodni položaj prehoda t_j, če obstaja povezava od položaja p_i k prehodu t_j. Podobno rečemo, da je položaj p_i izhodni položaj prehoda t_j, kadar obstaja povezava od prehoda t_j k položaju p_i. Označitev SPM je določena s porazdelitvijo žetonov po posameznih položajih. Žetone grafično predstavimo s pikami. Nek prehod je omogočen, kadar vsebuje vsak izmed njegovih vhodnih položajev vsaj en žeton. Če je prehod takojšen,

se izvrši takoj, če pa je časoven, se izvrši po eksponentno porazdeljenem naključnem času. Prehodna hitrost, pripisana temu prehodu, predstavlja parameter te porazdelitve. Izvršitev prehoda povzroči odvzetje enega žetona iz vseh vhodnih položajev in dodajanje enega žetona v vsak izhodni položaj. To povzroči novo razporeditev žetonov po položajih in s tem nastanek nove označitve. Vse označitve neke SPM tvorijo njeno dosegljivostno množico, ki jo lahko predstavimo z drevesom dosegljivosti. Graditi ga pričnemo pri začetni označitvi, ki predstavlja koren drevesa in poiščemo vse neposredno dosegljive označitve, ki jih dobimo z izvršitvijo prehodov, omogočenih z začetno označitvijo. Te označitve postanejo neposredni potomci korena. Ta postopek ponavljamo nad temi in vsemi nadaljnimi potomci, dokler se vsaka pot v drevesu ne konča z označitvijo, ki smo jo že predhodno srečali, ali pa z mrtvo označitvijo, to je tako, ki ne omogoča nobenega prehoda. Označitve iz tako dobljene dosegljivostne množice so lahko stabilne ali pa nestabilne. Za stabilne velja, da omogočajo le časovne prehode, medtem, ko je pri nestabilnih označitvah omogočen vsaj en takojšen prehod. Zato je čas trajanja nestabilnih označitev enak nič.

Dokazano je, da so SPM izomorfne s časovno zveznimi markovskimi verigami. Stabilne označitve iz dosegljivostne množice SPM sovpadajo s stanji iz prostora stanj ČZMV, prehodne hitrosti med stabilnimi označitvami SPM pa s časovnimi gostotami prehodov med stanji ČZMV. Zato je potrebno iz drevesa dosegljivosti izločiti vse nestabilne označitve in na novo preračunati prehodne hitrosti med stabilnimi. Možno pa je tudi, da že pri graditvi drevesa dosegljivosti upoštevamo samo stabilne označitve. Drevo dosegljivosti SPM z le stabilnimi označitvami predtavlja torej že kar diagram prehajanja stanj iskane ČZMV.

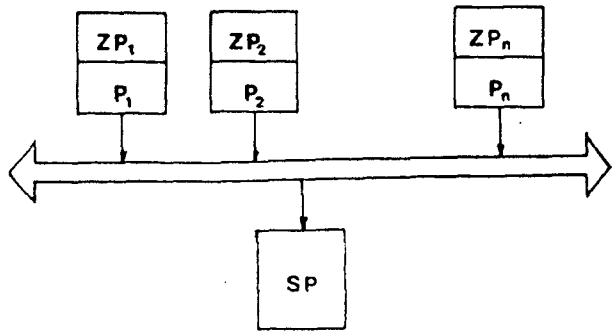
Postopek performančne analize nekega večprocesorskega sistema je z uporabo SPM precej olajšan. Od analitika zahteva samo definiranje topologije in parametrov SPM, nadaljni postopek gradnje drevesa dosegljivosti in stacionarne analize njemu izomorfne ČZMV pa je lahko popolnoma avtomatiziran. V naslednjem poglavju bomo pokazali celoten postopek performančne analize enostavne arhitekture večprocesorskega sistema.

5. Primer analize učinkovitosti večprocesorskega sistema

Za primer analizirajmo večprocesorski sistem s skupnim vodilom; model tega sistema prikazuje slika 2. Vsakemu procesorju je pridružen zasebni pomnilnik, skupni pomnilnik pa je zunanji vsem procesorjem, to pomeni, da je vsakemu izmed njih dostopen le preko skupnega vodila. Do konfliktnih situacij prihaja pri zaseganju skupnega vodila. Da bomo lahko dogajanje v tem sistemu obravnavali kot stohastični proces z markovsko lastnostjo, se mora režim delovanja modela podrejati naslednjim predpostavkam:

- Časi trajanja aktivnega stanja in stanja dostopa posameznih procesorjev so naključne spremenljivke z eksponentno porazdelitvijo in srednjimi vrednostmi
$1/\lambda_1, 1/\lambda_2 \ldots, 1/\lambda_p$ za aktivna stanja in
$1/\mu_1, 1/\mu_2, \ldots 1/\mu_p$ za stanja dostopa.
- Ko zahteva nek procesor dostop do skupnega pomnilnika, je ta dostop možen takoj (brez zakasnitve), če je skupno vodilo prosto.
- Če skupno vodilo ni prosto, preide procesor v fazo čakanja do sprostitve vodila.
- Ko procesor konča s stanjem dostopa, takoj (brez zakasnitve) sprosti skupno vodilo.

V analizi bomo predpostavljali popolno

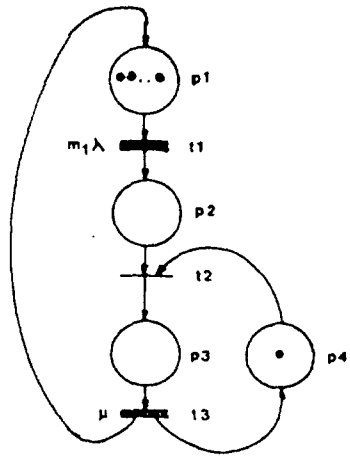


Slika 2. Arhitektura večprocesorskega sistema

simetrijo v sistemu, kar pomeni, da bosta parametra našega modela srednja vrednost dolžine aktivnega stanja $1/\lambda$ in srednja vrednost dolžine faze izmenjave $1/\mu$ enaka za vse procesorje. Razmerje med njima $\rho = \lambda/\mu$ pa imenujmo faktor obremenljivosti.

Točki 2. in 4. pomenita, da smo zanemarili čase, potrebne za dodeljevanje in sproščanje skupnega vodila. To dejanje lahko opravičimo z dejstvom, da so ti časi navadno veliko krajši od časov trajanja faz procesiranja in izmenjave. Možno pa je tudi, da čas dodeljevanja in sproščanja vodila prištejemo k času faze izmenjave.

Če torej sistem zadošča gornjim zahtevam, mu lahko priredimo stohastični proces z markovsko lastnostjo in opravimo performančno analizo na osnovi analize stacionarnega stanja prirejene markovske verige.



Slika 3. SPM večprocesorskega sistema

Analize se lotimo s konstruiranjem SPM, ki bo opisovala dogajanje v sistemu. Prikazuje jo slika 3. Število žetonov v položaju p_1 pomeni število procesorjev v aktivnem stanju, število žetonov v p_2 pomeni število procesorjev v stanju čakanja, žetoni v p_3 pomenijo procesorje v stanju dostopa, žeton v p_4 pa pomeni, da je skupno vodilo prosto. Prehod t_1 je časoven in predstavlja dogodek, ko želi eden izmed aktivnih procesorjev poseči v skupni pomnilnik. Ker je gostota teh dogodkov sorazmerna številu aktivnih procesorjev, je prehodna hitrost prehoda t_1 enaka $m_1\lambda$, kjer je m_1 število žetonov v p_1. Prehod t_2 je takojšen, saj dobi procesor dostop do skupnega pomnilnika takoj, če je le skupno vodilo prosto. Prehod t_3 je zopet časoven in ponazarja dogodek, ko nek procesor zaključi stanje dostopa in s tem sprosti skupno vodilo. Začetna označitev naj bo taka, da se v p_1 nahaja p žetonov (p je število procesorjev v sistemu), v p_4 en žeton, v p_2 in p_3 pa ni nobenega žetona. Ta začetna označitev ustreza stanju sistema, ko so vsi procesorji v

aktivnem stanju, skupno vodilo pa je prosto. Iz tako definirane stohastične Petrijeve mreže si zgradimo pripadajoče drevo dosegljivosti (slika 4). Pri graditvi drevesa smo takoj izpuščali nestabilne označitve in upoštevali le stabilne. Na primer. Iz označitve (p,0,0,1) pridemo v prvem koraku do označitve (p-1,1,0,1), ki je nestabilna, saj omogoča takojšen prehod t_2, zato takoj nastopi označitev (p-1,0,1,0), ki je stabilna in jo v drevesu upoštevamo.

$$
\begin{array}{lll}
P_0 = (p,0,0,1) & & \\
\quad\downarrow p\lambda & & \\
P_1 = (p-1,0,1,0) & \xrightarrow{\mu} & P_0 \\
\quad\downarrow (p-1)\lambda & & \\
P_2 = (p-2,1,1,0) & \xrightarrow{\mu} & P_1 \\
\quad\downarrow (p-2)\lambda & & \\
\quad\vdots & & \\
\quad\downarrow \lambda & & \\
P_p = (0,p-1,1,0) & \xrightarrow{\mu} & P_{p-1}
\end{array}
$$

Slika 4. Drevo dosegljivosti SPM

Drevo dosegljivosti SPM nam popolnoma določa ČZMV s katero bomo ponazorili dogajanje v našem večprocesorskem sistemu, saj predstavlja dosegljivostna množica prostor stanj ČZMV, prehodne hitrosti drevesa dosegljivosti pa so enake časovnim gostotam prehodov verige. Diagram prehajanja stanj tako dobljene markovske verige prikazuje slika 5.

Preden se lotimo reševanja markovske verige, se moramo odločiti za nek performančni pokazatelj, ki nam bo dal dovolj dobro informacijo o učinkovitosti večprocesorskega sistema. Za ta pokazatelj smo izbrali povprečno število aktivnih procesorjev (procesorjev v aktivnem stanju) in ga imenovali procesna moč sistema P. Izračunamo jo po enačbi

$$P = \Sigma_i (\pi_i n_a(i)), \quad i \in S, \tag{4.1}$$

kjer pomeni π_i stacionarno verjetnost stanja i ČZMV, $n_a(i)$ pa število aktivnih procesorjev v stanju sistema, ki je ponazorjen z i-tim stanjem ČZMV. Za izračun procesne moči rabimo torej stacionarne verjetnosti stanj verige, ki jih dobimo iz analize stacionarnega stanja ČZMV. To opravimo z rešitvijo sistema linearnih enačb, ki jih lahko zapišemo direktno iz diagrama prehajanja stanj po enačbah 3.11 in 3.12. Dobimo sistem p+1 enačb

$$-p\lambda\pi_0 + \mu\pi_1 = 0 \tag{4.2}$$

$$-((p-i)\lambda + \mu)\pi_i + (p-i+1)\lambda\pi_{i-1} + \mu\pi_{i+1} = 0, \quad i = 1 \ldots p-1$$

$$\pi_0 + \pi_1 + \ldots + \pi_p = 1.$$

Splošna rešitev tega sistema je naslednja:

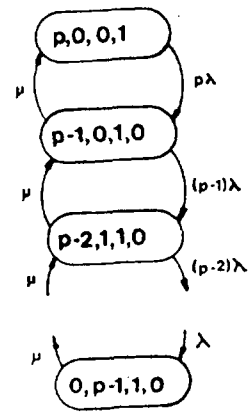
$$\pi_0 = (\Sigma_k (\rho^k (p!/(p-k)!)))^{-1}, \quad k = 0 \ldots p$$

$$\pi_i = \pi_0 \rho^i (p!/(p-i)!). \tag{4.3}$$

Procesna moč pa je enaka

$$P = (\Sigma_k (\rho^k n!/(n-k)!) - 1)/(\rho \Sigma_k (\rho^k n!/(n-k)!)), \quad k = 0 \ldots p \tag{4.4}$$

Rezultat analize učinkovitosti našega večprocesorskega sistema je torej enačba, ki podaja odvisnost procesne moči od števila procesorjev in obremenljivosti sistema. To pomeni, da lahko na podlagi rezultatov stohastične analize ocenimo učinkovitost sistema v različnem obsegu in pod različnimi delovnimi pogoji.



Slika 5. Diagram prehajanja stanj markovske verige

6. Zaključek

Rezultati predstavljene stohastične analize so dobljeni pod predpostavkami, navedenimi v poglavju 5. Te predpostavke so nam omogočile, da smo obravnavali dogajanje v večprocesorskem sistemu kot markovski proces. Čeprav se realni sistemi v splošnem ne podrejajo tem predpostavkam lahko uvidimo, da dajo markovski modeli navadno celo pesimistično oceno o učinkovitosti modeliranega sistema. Zakaj? V realnem sistemu ima porazdelitev časov dostopa navadno manjšo varianco v primeru z eksponentno porazdelitvijo. Tedaj je dejanska učinkovitost boljša, kot jo je napovedal naš model. Isto velja v primeru, ko časi aktivnih stanj in stanj dostopa niso za vse procesorje enaki. Iz tega lahko zaključimo, da dajejo rezultati, dobljeni na osnovi predstavljene stohastične analize, dovolj dobro oceno o učinkovitosti obravnavanega večprocesorskega sistema.

7. Literatura

1. Feller W., An Introduction to Probability Theory and Its Applications, Willey, New York, 1966.
2. Holliday M.A., Exact Performance Estimates for Multiprocessor Momory and Bus Interference, IEEE Transactions on Computers, Januar 1987.
3. Jamnik R., Verjetnostni račun, Mladinska knjiga, Ljubljana, 1971.
4. Karlin S., A First Course in Stochastic Processes, Academic Press, New York, 1975.
5. Marsan M.A. s sodelavci, Modeling Bus Contention and Memory Interference in a Multiprocessor System, IEEE Transactions on Computers, Januar 1983.
6. Marsan M.A. s sodelavci, Performance Models of Multiprocessor Systems, MIT Press, Cambridge, London, 1986.
7. Peterson J.L., Petri Net Theory and the Modeling of Systems, Prentice-Hall, Englewood Cliffs, 1981.
8. Vidav I., Višja matematika II, DZS, Ljubljana, 1975.
9. Zuberek W.M., Timed Petri Nets and Preliminary Performance Evaluation, Proc. of the 7th Annual Symposium on Computer Architecture, La Baule, 1980.