Poglavje 1

Uvod v analizo zmogljivosti računalniških sistemov

Pod pojmom *analize zmogljivosti računalniških sistemov* smatramo nabor postopkov, ki nas privedejo do *kvalitativne* ali *kvantitativne ocene* zmogljivosti (angl. *performance*) delovanja opazovanega računalniškega sistema.

Kompleksnost, *zmogljivost*, *arhitektura* in *namembnost* računalniških sistemov so se v zadnjih petdesetih letih hitro spreminjale. V splošnem so računalniški sistemi postajali vse bolj kompleksni in vse bolj zmogljivi, njihova namembnost pa se je s časom spreminjala. Kronološko računalniške sisteme glede na njihovo *namembnost*, *arhitekturo* in *način pojavljanja na trgu* razdelimo v naslednje kategorije:

- doba večuporabniških „mainframe“ računalnikov (angl. *mainframe computing*): ta doba je trajala od šestdesetih let pa do sredine devetdesetih let prejšnjega stoletja; storitve posameznega „mainframe“ računalnika je istočasno lahko uporabljalo veliko število ljudi (angl. *one computer - many people*), tovrstni sistemi pa so bili izredno kompleksni in dragi;

- doba osebnih „namiznih“ računalnikov (angl. *desktop computing*): ta doba je trajala od sredine osemdesetih let, pa vse do danes; omenjena doba se končuje zaradi vse večje odvisnosti naših *končnih delovnih točk* (namiznih in prenosnih računalnikov) od storitev spletnega omrežja in strežnikov, ki se v omrežju nahajajo; tako se v zadnjem desetletju na vseh področjih računalništva uveljavi koncept odjemalca in strežnika (angl. *client - server*); z vse večjo povezanostjo z atraktivnimi omrežnimi storitvami in s tem posredno odvisnostjo od njih, končne delovne točke počasi izgubljajo predhodni primarni funkciji *obdelave* in *hrambe* podatkov; obe funkcionalnosti z vidika resursov vse bolj zaupamo omrežnim rešitvam; tipičen primer tovrstnih rešitev predstavljajo oblačne storitve (angl. *cloud services*, *cloud computing*); funkcijo odjemalca ali končne točke vse bolj prevzemajo manjše naprave kot so pametni telefoni, tablični računalniki in

„lahki“ odjemalci (angl. *thin client*); pod slednje smatramo računalniške naprave brez zmožnosti lokalnega trajnega pomnjenja podatkov [1]; tako doba namiznega računalništva počasi tone v pozabo; v tej dobi posamezni namizni računalnik istočasno lahko uporablja le en človek (angl. *one computer - one person*);

- doba „vseprisotnega računalništva“ (angl. *pervasive computing*): ta doba se začne sredi devetdesetih let prejšnjega stoletja, ko računalniški sistemi začnejo vstopati tudi v sisteme, kjer jih predhodno nismo zasledili; tipični primeri tovrstnih rešitev so digitalizacija bele tehnike, digitalizacija osebnih vozil, digitalizacija medicinskih naprav, v prihodnosti pa lahko na tem področju pričakujemo na osnovi paradigme kot je IoT (angl. *Internet of Things*) prodor tovrstnih rešitev v vse pore našega življenja (trenutno so aktualna področja inteligentnih oblačil, sledenja prehrambene preskrbovalne verige, inteligentnih zdravil itd.); „vseprisotni računalniki“ so primarno namenjeni zagotavljanju storitev posamezniku (angl. *one person - many computers*), ali pa tudi več ljudem;

Glede na hiter razvoj računalniških sistemov so se skozi čas spreminjale tudi *metrike* ali *mere*, s katerimi so se merile zmogljivosti računalniških sistemov. Več o zmogljivostnih ali performančnih metrikah povemo v naslednjih razdelkih.

1.1 Vrste metrik za ocenjevanje zmogljivosti računalniških sistemov

V uvodnem delu pričujočega poglavja smo prvič omenili pojma *zmogljivostne metrike* in *zmogljivostne mere*, ki definirata, kaj v računalniškem sistemu v okviru zmogljivostne analize pravzaprav ocenjujemo ali merimo. Zmogljivostne metrike za ocenjevanje zmogljivosti računalniških sistemov po viru [2] delimo na sledeče skupine:

- *preštevalne metrike*: omenjene metrike preštevajo dogodke (npr. prešteje pakete, ki v določenem časovnem intervalu prispejo na usmerjevalnik, ali število zahtev, ki v določenem časovnem intervalu prispejo na spletni strežnik itd.);

- *časovne metrike*: omenjene metrike merijo čase trajanja opazovanih procesov (npr. čas izvajanja programa, čas za pripravo odgovora na spletnem strežniku, čas potovanja paketa od izvora do ponora itd.);

- *vrednostne metrike*: omenjene metrike vrednostno ocenjujejo nek opazovani parameter (npr. frekvenco delovanja procesorja, velikost dinamičnega pomnilnika itd.);

Naslednja možna delitev metrik je njihovo razvrščanje glede na vrednosti, ki jih metrika lahko zavzame. Tovrstna delitev je po viru [3] sledeča:

- *manjše je boljše*: upravljalci sistema si prizadevajo doseči čim manjše vrednosti tovrstnih metrik; primer takšne metrike je čas odgovora sistema;
- *večje je boljše*: upravljalci sistema si prizadevajo doseči čim višje vrednosti tovrstnih metrik; primer takšne metrike je prepustnost sistema;
- *srednje je najboljše*: v tem primeru mejne vrednosti metrike niso zaželjene; primer takšne metrike je obremenitev omrežja ETHERNET; pri majhni obremenitvi omrežje ni izkoriščeno, pri velikih obremenitvah omrežja pa je uporabniška izkušnja slaba, saj se promet odvija upočasnjeno; v mnogih praktičnih primerih je željena vrednost tovrstne metrike približno $\frac{2}{3}$, pri kateri uporabnik še ne občuti degradiranega ali upočasnjenega delovanja sistema;

Koncept metrik po opisani delitvi je predstavljen na sliki 1.1, povzeti po viru [3].

Slika 1.1: Razredi zmogljivostnih metrik: (a) manjše je boljše, (b) večje je boljše, (c) srednje je najboljše [3].

1.2 Lastnosti dobrih metrik

Ključno vprašanje pri rangiranju različnih računalniških sistemov glede na njihove zmogljivosti je vprašanje izbire ustrezne metrike, ki nam bo podala najbolj verodostojno oceno primerjave med opazovanimi sistemi. Dobre metrike morajo imeti *kvantitativno bazo*, omogočati morajo *primerjavo med sistemi*, navajati morajo na *ustrezne zaključke* in morajo biti *razumljive* uporabnikom. Preostale lastnosti dobrih metrik so po viru [2] sledeče:

- *linearnost*: omenjena lastnost predpostavlja linearno odvisnost med pridobljeno vrednostjo metrike in zmogljivostjo sistema; tako se npr. povečanje zmogljivosti delovanja sistema za določen delež odraža v zvečanju (ali zmanjšanju) vrednosti metrike za isti delež; kot primer obravnave linearnosti navedimo primer hitrosti izvajanja programa; v primeru, da na nivoju procesnih resursov podvojimo zmogljivost procesiranja instrukcij, pričakujemo, da se bodo programi izvajali dvakrat hitreje in da bo čas izvajanja programa za polovico krajši; linearnostna narava metrike je pomembna predvsem z vidika lažje človeške percepcije; pri tem poudarjamo, da ni nujno, da je metrika linearna, je pa v primeru linearnosti dosti bolj razumljiva; kot primer nelinearne metrike navajamo logaritemsko metriko (uporabljamo jo npr. za merjenje jakosti zvoka);

- *zanesljivost*: metrika je zanesljiva, ko njena pridobljena vrednost venomer vodi do pravilne ocene delovanja primerjanih sistemov z zmogljivostnega vidika; tipičen primer nezanesljive metrike je MIPS metrika (angl. *millions of instructions executed per second* - MIPS), ki meri število izvedenih instrukcij procesorja v milijonih instrukcij na sekundo; omenjena metrika nam v mnogo primerih ne zagotavlja, da se bo program na sistemu z višjo vrednostjo metrike MIPS v resnici izvajal hitreje;
- *ponovljivost*: metrika je ponovljiva, če ob ponavljanju istega eksperimenta pridobivanja vrednosti metrike pridobimo venomer njeno isto vrednost; slednja značilnost predpostavlja, da je metrika *deterministične narave*, čemur v praksi venomer ni tako; predpostavimo, da merimo število prispelih paketov na opazovani usmerjevalnik; glede na časovno spremenljiv promet v omrežju omenjena metrika ni deterministična, temveč je stohastične narave, s katero ponazorimo naključnost dogajanja in pa izražanje časovno spremenljivih trendov;
- *enostavnost*: metrika je enostavna, če je njena uporaba enostavna; enostavnost uporabe glasi na način pridobivanja vrednosti metrike (npr. štetja prispelih paketov na usmerjevalnik); pridobivanje vrednosti metrik je lahko izredno enostavno, lahko pa tudi izredno kompleksno;
- *konsistentnost*: konsistentna metrika je tista, katere definicija in enote so enake ne glede na opazovani sistem analize in njegovo konfiguracijo; kot primer nekonsistentne metrike lahko zopet navedemo metriko MIPS, katere nekonsistentnost izhaja iz neenakih instrukcijskih naborov na različnih sistemih;
- *neodvisnost*: dobra metrika naj bi bila neodvisna od proizvajalca sistema, saj so v preteklosti proizvajalci vsiljevali tržišču svoje lastne metrike, po katerih so se njihovi sistemi najbolje izkazali;

1.3 Tradicionalne metrike za ocenjevanje zmogljivosti računalniških sistemov

V dobah „mainframe“ in „namiznih“ računalnikov so se metrike razvijale postopoma z naraščanjem zmogljivosti sistemskih resursov. Glede na to, da se omenjeni dobi v kontekstu široke uporabe iztekata, omenjene metrike poimenujemo za „tradicionalne“. Slednje so naštete v naslednjih razdelkih in povzete po viru [2].

1.3.1 Frekvenca delovanja procesorja

Frekvenca delovanja procesorja ali *takt ure* je ena od najbolj razširjenih zmogljivostnih metrik. Običajno predpostavljamo, da bo izvajanje programskih bremen ob zvečanju frekvence delovanja procesorja hitrejše. Marsikdaj slednja predpostavka ne velja. Slabosti metrike so sledeče:

- definicija metrike ne definira natančno, kaj se izvede v enem urinem ciklu delovanja procesorja;
- definicija metrike ignorira kompleksne interakcije s pomnilnikom in I/O segmentom;
- definicija ne upošteva dejstva, da procesor običajno ne predstavlja ozkega grla v domeni procesiranja odziva sistema;

Glede na opisane slabosti metrike običajno zvečanje takta ure ne povzroči linearne pospešitve procesiranja. Metrika je ponovljiva, enostavna, konsistentna in neodvisna, ni pa linearna in zanesljiva.

1.3.2 MIPS metrika

MIPS (angl. *millions of instructions executed per second*) metrika predstavlja število milijonov instrukcij, ki jih sistem izvede na sekundo. Sama vrednost metrike je odvisna od nabora instrukcij, vrstnega reda izvajanja instrukcij in prisotnosti vejalnih instrukcij. Metrika je ponovljiva, enostavna in neodvisna, ni pa linearna, zanesljiva in konsistentna. Temeljni problem metrike je v tem, da ne izraža neposredno količine izvedenega deleža sistemskega bremena (angl. *amount of computation*) v povezavi z izvedbo posamezne instrukcije. Glede na povedano se MIPS metrika vse manj uporablja za izvedbo realističnih zmogljivostnih analiz. Metrike se je glede na njene opisane slabosti oprijelo ime „brezpomenski indikator zmogljivosti“ (angl. *meaningless indicator of performance* - MIPS).

1.3.3 MFLOPS metrika

MFLOPS (angl. *millions of floating point operations executed per second*) metrika predstavlja število milijonov instrukcij v plavajoči vejici, ki jih sistem izvede na sekundo. Metrika je do neke mere boljša od MIPS metrike, saj so instrukcije (operacije) v plavajoči vejici specifičnejše od splošnih instrukcij, njene slabosti pa so podobne slabostim MIPS-a, povrh vsega pa je metrika neuporabna za problemsko domeno aplikacij, v katerih je število operacij v plavajoči vejici majhno ali celo nično. Primera takšnih segmentov opazovane večje aplikacije sta segmenta iskanja ali sortiranja. Tudi pri MFLOPS metriki se soočamo z dejstvom, da se nabori instrukcij za izvajanje operacij v plavajoči vejici med sistemi drastično razlikujejo glede na kompleksnost posameznih instrukcij, kar vodi do tega, da ena instrukcija v kompleksnem naboru lahko opravi delo več instrukcij iz enostavnejšega nabora instrukcij. Metrika je tako ponovljiva, enostavna in neodvisna, ni pa linearna, zanesljiva in konsistentna.

1.3.4 SPEC metrike

SPEC metrike so plod dela neprofitnega združenja z isto kratico (angl. *standard performance evaluation cooperative* - SPEC). Cilj SPEC metrik je pridobivanje

zmogljivostnih ocen delovanja sistema pri nekem tipičnem vzorčnem bremenu. Metodologija vrednotenja metrik temelji na izvajanju množice testnih tipičnih uporabniških programov. Slednje imenujemo za „benchmark“ programe. Za izvajanje vsakega od tipičnih programov nam SPEC orodja izmerijo čas izvajanja, normalizirani izmerjeni časi pa se primerjajo z izvajalnimi časi na referenčnih arhitekturah računalniških sistemov. Omenjene metrike imajo značilnosti ponovljivosti, enostavnosti in konsistentnosti, nimajo pa značilnosti linearnosti in zanesljivosti. Sporna je tudi značilnost neodvisnosti, saj je SPEC združenje izpostavljeno stalnim pritiskom proizvajalcev po „ustreznem“ oblikovanju vzorčnih bremenskih programov, ki bi izpostavljali prednosti njihovih sistemov, istočasno pa proizvajalci prilagajajo svoje sisteme na vzorčne bremenske programe. Tipični primeri tovrstnih prilagajanj v preteklosti so bili postopki optimizacij, ki so jih izvajali prevajalniki izvorne kode.

V splošnem so „benchmark“ programi usmerjeni na meritve zmogljivosti vseh vitalnih delov računalniškega sistema. Mednje npr. sodijo meritve hitrosti izvajanja različnih procesorskih ukazov (npr. SPEC CPU2017) in meritve hitrosti izvajanja transakcij s trajnim pomnilnim medijem (npr. SPEC SFS2014).

Več o delovanju združenja SPEC si lahko bralec ogleda na spletni strani navedeni v viru [4]. Na omenjeni spletni strani so na razpolago tudi „benchmark“ programi za vrednotenje novejših metrik na modernejših arhitekturah in storitvah kot so oblačne arhitekture, poštni strežniki, datotečni strežniki, virtualizirane arhitekture itd.

1.3.5 Izvajalni čas

Izvajalni čas (angl. *execution time*) je metrika, ki predstavlja čas izvajanja računalniškega programa. Glede na nivo opazovanja sistema se metrika lahko vrednoti tudi z vidika časa izvajanja zahteve, storitve, transakcije itd. Metrika je tipa „manjše je, boljše je“. Osnova za pridobivanje vrednosti metrike je merjenje časa izvajanja. Slednje običajno ni najbolj enostavno, saj se nam lahko v meritev prikradeta *sistemska napaka* in *šum*. Z vidika same izvedbe lahko izvajanje merimo na nivoju samega programa (v samem programu) ali na nivoju operacijskega sistema (izven programa). Primer C-jevske kode merjenja časa izvajanja segmenta kode v samem programu povzet po viru [2] je naveden v izpisu 1.1, merjenje izven programa pa je pogojeno z razpoložljivimi funkcijami operacijskega sistema in je odvisno od opazovanega sistema.

Listing 1.1: Primer opremljanja programa z merilnim segmentom [2].

```
main()
{ int i;
  float a;

        init_timer();

        /* Read the starting time */
        start_count = read_count;
```

Značilnost	linearn.	zaneslj.	ponovlj.	enost.	konsist.	neodvis.
Takt ure	NE	NE	DA	DA	DA	DA
MIPS	NE	NE	DA	DA	NE	DA
MFLOPS	NE	NE	DA	DA	NE	DA
SPEC	NE	NE	DA	DA	DA	NE
Izvajalni čas	NE	DA	DA	DA	DA	DA

Tabela 1.1: Primerjava tradicionalnih zmogljivostnih metrik.

```
        /* Stuff to be measured */
        for (i=0; i<1000; i++)
          { a = i * a / 10;}

        /* Read the ending time */
        end_count = read_count;

        elapsed_time=(end_count-start_count)*clock_cycle;
}
```

Predhodno smo omenili pojem sistemske napake meritve. V primeru izvorne kode v zapisu 1.1 bi se nam le ta lahko prikradla v primeru večuporabniškega ali večopravilnega operacijskega sistema, kjer bi bilo izvajanje izseka programa izvajano v več korakih, zaradi občasnega dodeljevanja sistemskih resursov drugim uporabnikom/programom. Glede na povedano moramo biti pri pridobivanju izvajalnega časa pozorni na to, kaj v resnici sploh merimo (število procesnih rezin izvajanja, realni čas izvajanja z vidika uporabnika, ki je običajno daljši itd.). Tudi če sistem ni večopravilen ali večuporabniški, se lahko izmerjeni časi izvajanja istega programa ali izseka kode razlikujejo glede na ostala sistemska opravila (funkcije operacijskega sistema v ozadju, stanja predpomnilnika, zapolnjenosti dinamičnega pomnilnika itd.). Glede na povedano lahko napravimo zaključek, da je lahko izvajalni čas programa ali izseka kode do neke mere nedeterminističen. Iz tega vidika je smiselno meritve izvajalnega časa ponoviti večkrat in za rezultat meritev smatrati povprečen čas ter maksimalni in minimalni čas izvajanja. V primeru večkratnih ponovitev meritev vrednosti metrike je tako metrika zanesljiva, ponovljiva, enostavna, konsistentna ter neodvisna, ni pa linearna. S tega zornega kota metriko smatramo kot najboljšo od tradicionalnih zmogljivostnih metrik.

1.3.6 Povzetek tradicionalnih zmogljivostnih metrik

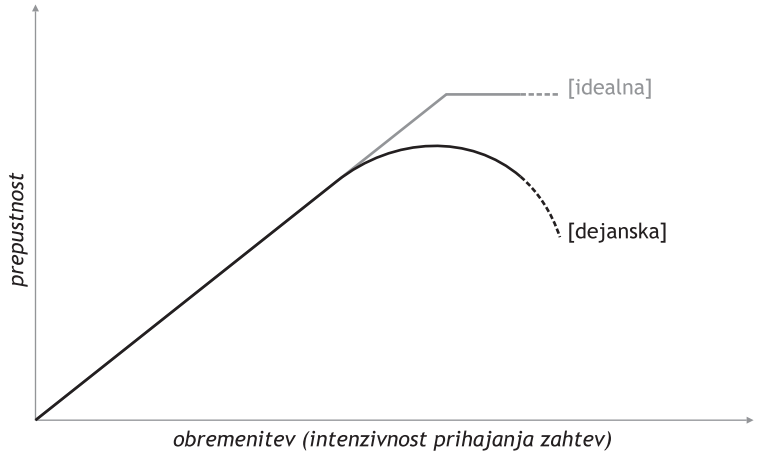
V tabeli 1.1 je prikazana primerjava do sedaj obravnavanih tradicionalnih zmogljivostnih metrik glede na izpolnjenost kriterijev „dobre“ metrike. Poleg naštetih metrik, poznamo še množico drugih tradicionalnih metrik. Le te so sledeče [2]:

- *čas odgovora sistema* (angl. *system response time*): gre za čas, ki mine od izdaje zahteve uporabnika do njegovega prejema odgovora na zahtevo; metrika se uporablja pri analizi zmogljivosti transakcijskih procesnih sistemov (angl. *on line transaction processing systems*); mednje sodijo elektronski bančni sistemi, elektronski plačilni sistemi, sistemi bančnih avtomatov, računalniški sistemi prodaje letalskih kart itd.;

- *prepustnost sistema* (angl. *system's throughput*): prepustnost R je po viru [3] definirana kot število zahtev N, ki jih računalniški sistem postreže v opazovanem času t (1.1); če sistem ni v nasičenju in ga opazujemo dovolj dolgo, je prepustnost enaka intenzivnosti prihajanja zahtev v sistem, kar pomeni, da se vsaka prispela zahteva v nekem končnem času obdela in zapusti sistem; sistem je v nasičenju, ko je vsaj eden od strežnikov sistema maksimalno obremenjen, kar ima za posledico rast čakalnih vrst preko vseh meja in s tem v praksi izgubljanje zahtev;

$$R = \frac{N}{t} \tag{1.1}$$

  s povečevanjem intenzivnosti prihajanja zahtev se povečuje tudi prepustnost sistema, vendar največ do neke maksimalne - teoretične ali *idealne prepustnosti*; na sliki 1.2 je prikazana *dejanska prepustnost*, ki s povečeva-



Slika 1.2: Idealna in dejanska prepustnost.

  njem vhodne intenzivnosti do nasičenja narašča, pri nasičenju pa doseže maksimalno prepustnost, ne doseže pa idealne; v realnih sistemih se nikoli ne doseže idealne prepustnosti, ampak le dejanska prepustnost; ker je sistem pri povečevanju vhodne intenzivnosti dodatno obremenjen, se prepustnost zmanjša, kar vodi do manjše učinkovitosti sistema; zato je potrebno vedno paziti, da sistem ni preobremenjen s številom zahtev, saj njegova prepustnost pri preobremenjenosti pade;

- *pasovna širina prenosa podatkov* (angl. *bandwidth*): metriko uporabljamo za ocenjevanje hitrosti prenosa podatkov v računalniških omrežjih, merimo pa jo običajno v bitih na sekundo (bs, bps), ali večjih enotah (npr v Mbs ali Gbs);

- *velikost dinamičnega pomnilnika*; omenjena metrike je vrednostne narave, ima pa poleg frekvence procesorja običajno velik osnovni vpliv na hitrost delovanja sistema;

Poleg naštetih metrik lahko omenimo še ocenjevalne metrike z vidika *interoperabilnosti* sistemov. V slednjem primeru računalniške sisteme ali njihove storitve vrednotimo predvsem z njihovo zmožnostjo semantične interoperabilnosti.

1.4 Metode za analizo zmogljivosti računalniških sistemov

Metode analize zmogljivosti računalniških sistemov lahko izvajamo po virih [2], [3] na tri načine. Ti so sledeči:

- *pristop z meritvami*: najbolj verodostojne rezultate zmogljivostne analize dobimo na osnovi pristopa z meritvami, ki jih izvajamo na realnem (obstoječem) sistemu; prvi problem meritev je kaj storiti, če sistem še ne obstaja, ali do njega nimamo dostopa; drugi problem meritev je v variabilnosti konfiguracije sistema; večina sistemov lahko namreč nastopa v različnih konfiguracijah (npr. različne možnosti izbire velikosti dinamičnega pomnilnika); na tem mestu se postavlja vprašanje, koliko časa porabimo za spremembo konfiguracije in ali je sprememba glede na razpoložljive resurse sploh možna;

- *simulacijski pristop*: slednji pristop nam sicer vzame kar nekaj časa, da vzpostavimo *model sistema*, ko pa imamo model, je v njem enostavno spreminjati variabilne parametre konfiguracije, kar je njegova prednost pred pristopom z meritvami; slabost simulacijskega pristopa je v tem, da v model običajno ne vgradimo vseh manjših detajlov sistema, tako da je model na nek način le posplošitev opazovanega sistema; v tem smislu pri samih simulacijah delovanja sistema na osnovi modela izgubljamo na natančnosti rezultatov;

- *analitični pristop*: slednji pristop temelji na matematičnih zakonitostih *teorije strežbe*; osnovne entitete omenjene teorije so *zahteve*, *strežniki* in pa *čakalne vrste*, v katere se uvrščajo zahteve v primeru, da so strežniki zasedeni; analitični pristop je uporaben predvsem takrat, ko na sistemu ne moremo izvajati meritev in bi bila izgradnja ustreznega modela sistema časovno prepotratna; več o matematičnih osnovah teorije strežbe si bralec lahko ogleda v delu [5];

Značilnost	Analitični pristop	Simulacijski pristop	Merilni pristop
Fleksibilnost	Visoka	Visoka	Nizka
Cena	Nizka	Srednja	Visoka
Prepričljivost	Nizka	Srednja	Visoka
Natančnost	Nizka	Srednja	Visoka

Tabela 1.2: Primerjava pristopov k zmogljivostni analizi.

Primerjava pristopov analize zmogljivosti je prikazana v tabeli 1.2, povzeti po viru [2]. Primerjava je izvedena na osnovi fleksibilnosti, cene, ter prepričljivosti in natančnosti rezultatov posameznega pristopa.

S pojmom fleksibilnosti ocenimo možnost spreminjanja variabilnosti konfiguracije (parametrov) sistema, s pojmom cene pa finančni vložek, ki je potreben za izvedbo analize in je pogojen z nakupom specifične opreme (simulacijskega programskega orodja ali merilne opreme) in s časovnim vložkom potrebnega dela. Slednjega običajno ocenjujemo z ekvivalenti polne zaposlitve posameznega človeka (angl. *full-time equivalent* - FTE). Pojem prepričljivosti (angl. *believabilty*) rezultatov izbrane vrste analize izraža stopnjo zaupanja, ki jo izkazujemo do pridobljenih rezultatov analize. Pod pojmom natančnosti (angl. *accuracy*) smatramo kako natančni so doseženi rezultati analize v primerjavi z zmogljivostjo opazovanega realnega sistema.

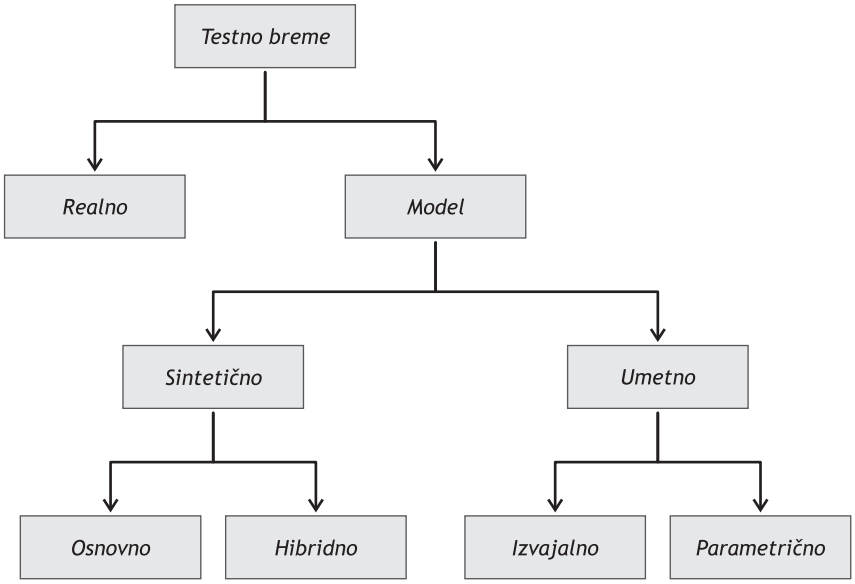
1.5 Breme računalniškega sistema

Pod pojmom *bremena* (angl. *computer workload*) računalniškega sistema smatramo *število* in *vrsto zahtev*, ki vstopajo v opazovani računalniški sistem, da bi bile postrežene. Tako za zahteve lahko smatramo pakete (v primeru računalniških omrežij), elektronske pošte (v primeru poštnega strežnika), nakupe (v primeru elektronske trgovine), programe (v primeru večopravilnega računalniškega sistema), telefonske klice in sporočila (v primeru baznih postaj mobilne telefonije) itd.

Število vstopajočih zahtev natančneje ponazorimo s pojmom *intenzivnosti porajanja zahtev* λ, ki jo merimo v številu prispelih zahtev na časovni interval (npr. število prispelih paketov na usmerjevalnik na sekundo). Poleg številske opredelitve spremenljivke λ je ključen dejavnik pri njej tudi porazdelitev medprihodnih časov, ki je lahko deterministična (medprihodni čas so enaki) ali statistično porazdeljena (npr. eksponentno ali normalno porazdeljena).

V predhodnem razdelku smo govorili o različnih vrstah analize zmogljivosti. Ne glede na vrsto analize smo venomer soočeni z določitvijo bremena, ki služi kot osnova za izvajanje analize. V tem primeru govorimo o *testnem bremenu*, ki ga uporabljamo za potrebe analize zmogljivosti. Testno breme je lahko *realno breme* ali pa *model bremena* (glej sliko 1.3, povzeto po [3]).

Realno breme je tisto, ki obremenjuje sistem v realnem času. Ker je realno breme stohastično pogojeno, je s tem neponovljivo, kar je pri ugotavljanju zmogljivosti moteče. Druga slaba lastnost realnega bremena je njegova pogo-



Slika 1.3: Vrste testnih bremen [3].

sta nedosegljivost (realno breme je npr. lahko poslovna skrivnost naročnika zmogljivostne analize). Pri ugotavljanju zmogljivosti želimo ponavljati meritve ali simulacije pri različnih parametrih - konfiguracijah sistema, zato je nujno potrebno, da je breme *ponovljivo* in da so eksperimenti simulacij ali meritev izdelani pod enakimi pogoji. Zato je boljša izbira za izvajanje eksperimentov analize model bremena, ki ni odvisen od realnega stohastično pogojenega okolja izvora zahtev bremena. Tako realno breme ni najboljša izbira za testno breme.

Modeli bremena so lahko *sintetični* ali *umetni*. Sintetična bremena so sestavljena iz podatkov o realnih bremenih (aplikacijah), pridobljenih na osnovi meritev, umetna pa na osnovi domnev o realnih bremenih (aplikacijah). *Osnovno sintetično breme* je izdelano na osnovi meritev pri enem samem bremenu (eni aplikaciji) in je namenjeno ugotavljanju zmogljivosti v izredno omejenih pogojih. Primer takšnega bremena je breme, izdelano na osnovi urejevalnika besedil, kot je na primer Microsoft Word. *Hibridno sintetično breme* je splošnejše in je sestavljeno iz najbolj reprezentativnih delov različnih bremen (aplikacij). Primer takšnega hibridnega bremena so testi za ugotavljanje zmogljivosti, ki jih pripravlja skupina SPEC. *Izvajalna umetna bremena* so zbirke ukazov, ki niso povezane z realnim bremenom. Nastajajo zaradi želje razvijalcev po enostavnih bremenih za primerjanje sistemov. Uporabna so zgolj za ugotavljanje zmogljivosti v zelo omejenih primerih. Kot zgled tega navedimo analizo zmogljivosti dostopa do določenega registra, do katerega želijo razvijalci doseči čim hitrejši dostop. Pri tem se postavlja vprašanje, kako vpliva hitrost dostopa do omenjenega registra na hitrost delovanja celotnega sistema v realnih pogojih. *Parametrično umetno breme* je podano zgolj s parametri. Primer takšnega

bremena je na primer lingvistični zapis 'intenzivnost strežbe je 10 zahtev na sekundo, porazdeljenih po eksponenti verjetnostni porazdelitvi'. Pri simulaciji in analitičnem reševanju problemov je uporaba umetnega parametričnega bremena dobra izbira, saj nam poenostavi delo. Več o bremenih računalniških sistemov si lahko bralec prebere v delu [3].

Literatura

[1] "Thin clients." `http://www.webopedia.com/DidYouKnow/Hardware_Software/thin_client.asp`, Februar 2017.

[2] D. J. Lilja, *Measuring computer performance*. Cambridge University Press, 2000.

[3] N. Zimic and M. Mraz, *Temelji zmogljivosti računalniških sistemov*. Založba FE in FRI, 2006.

[4] "SPEC Corporation." `https://www.spec.org/`, Februar 2018.

[5] M. Mraz and M. Moškon, *Modeliranje računalniških omrežij*. Založba FE in FRI, Ljubljana, 2012.