Buku Pegangan Kuliah

Analisa Kinerja Sistem

Jusak Irawan, PhD

Sekolah Tinggi Manajemen Informatika
dan Teknik Komputer Surabaya
STIKOM

Last updated: July 10, 2007

Chapter 0

Kontrak Perkuliahan

Nama Matakuliah	: Analisa Kinerja Sistem
Kode Matakuliah	:
Jumlah SKS	: 3 sks
Jadwal	: Senin, Rabu, Jum'at (07.00 - 09.30 am)
Penilaian	: - 30% Quiz (Setiap Senin)
	: - 40% Ujian Akhir Semester
	: - 30% Tugas (Setiap Jum'at)

Tujuan:

1. Mahasiswa mampu melakukan analisa terhadap unjuk kerja sebuah sistem dan membandingkannya dengan sistem lain yang sejenis.
2. Mahasiswa mampu merepresentasikan hasil pengolahan data dalam bentuk simulasi dan melakukan analisa berdasarkan simulasi tersebut.

Note:

Materi Perkuliahan

Materi perkuliahan meliputi:

1. Introduction to System Performance Analysis
2. MATLAB programming
3. Workloads

4. Workload characterization techniques
5. Monitors
6. Capacity Planning and Benchmarking
7. Data Representation
8. Ratio Games
9. Comparing system using sampled data
10. Regression Models
11. Queuing Theory
12. Web Services.

Bibliography

[1] Jain, Raj. *The Art of Computer System Performance Analysis,* John Wiley & Sons, Inc. New York, USA, 1991.

[2] Hennessy, J.L., Goldberg D. and Patterson, D.A. *Computer Architecture A Quantitative Approach. Third Edition,* Elsevier Science & Technology 2002.

[3] Lazowska, E.D., Zahorjan, J., Graham, G.S., and Sevcik, K.C. *Quantitative System Performance,* Prentice Hall, Inc. Englewood Cliffs, New Jersey, 1984.

Chapter 1

Pendahuluan

Evaluasi Unjuk Kerja

Note:

- Performance (unjuk kerja) merupakan kriteria utama dalam desain dan implementasi sistem komputer.
- Tujuan utama bagi para enjinier adalah untuk memperoleh unjuk kerja paling tinggi dengan biaya tertentu.
- Seorang enjinier harus memiliki pengetahuan dasar dalam hal terminologi dan tehnik evaluasi terhadap unjuk kerja.

Setelah mengikuti kuliah ini, anda diharapkan dapat mengetahui hal-hal berikut ini:

1. Memilih tehnik-tehnik evaluasi, ukuran suatu unjuk kerja dan beban kerja (workload) dari sebuah sistem yang sesuai.
2. Melakukan pengukuran unjuk kerja secara benar.
3. Menggunakan metoda-metoda statistik untuk melakukan komparasi terhadap beberapa alternatif.
4. Melakukan desain terhadap pengukuran dan simulasi unjuk kerja dengan usaha sekecil-kecilnya.

5. Melakukan simulasi secara benar.

Evaluasi Unjuk Kerja Sebagai Seni

Note:

- Melakukan evaluasi unjuk kerja bak seorang seniman. Hasil dari sebuah evaluasi yang disebut sukses tidak tidak dapat dihasilkan secara mekanis.
- Setiap evaluasi membutuhkan pengetahuan yang dalam tentang sistem yang dimodelkan dan pemilihan secara hati-hati terhadap metodologi, workload serta tool yang akan digunakan.
- Sebagai contoh perhatikan tabel hasil pengukuran *throughput* dari dua buah sistem di bawah ini:

Table 1.1: Throughput dalam transaksi per detik

Sistem	Workload 1	Workload 2
A	20	10
B	10	20

Berdasarkan Tabel 1.1 ada tiga cara yang dapat dilakukan untuk membandingkan kedua buah sistem. *Pertama,* dengan mengambil unjuk kerja rata-rata dari kedua buah beban kerja (workload). Perhatikan hasil analisa tersebut pada Tabel 1.2, dimana kedua buah sistem terlihat memiliki unjuk kerja yang sama. *Kedua,* dengan cara menghitung rasio kedua buah workload dengan menggunakan sistem B sebagai dasar. Hasil analisa ditunjukkan dalam Tabel 1.3, dimana sistem A terlihat lebih baik daripada sistem B. *Ketiga,* dengan cara menghitung rasio kedua buah workload dengan menggunakan sistem A sebagai dasar. Hasil analisa ditunjukkan dalam Tabel 1.4, dimana sistem B terlihat lebih baik daripada sistem A.

Table 1.2: Perbandingan Menggunakan Rata-Rata

Sistem	Workload 1	Workload 2	Rata-rata
A	20	10	15
B	10	20	15

Table 1.3: Rasio Menggunakan Sistem B Sebagai Dasar

Sistem	Workload 1	Workload 2	Rata-rata
A	2	1.5	1.25
B	1	1	1

Kesalahan-Kesalahan Umum Dalam Hal Evaluasi Unjuk Kerja

Note:

1. *Tidak memiliki tujuan (goal):* Setiap model dari sebuah sistem harus dibangun dengan tujuan tertentu. Satuan, workload dan metodologi sangat bergantung pada tujuan tersebut. Karena itu hal yang paling penting adalah mengidentifikasi permasalahan yang akan diselesaikan terlebih dahulu.

2. *Tujuan tidak jelas.*

3. *Pendekatan yang tidak sistematis:* Seringkali para analis menggunakan pendekatan yang tidak sistematis dalam memilih parameter dari sistem, faktor, satuan dan workload karena pemilihan tersebut dilakukan secara sembarangan. Pada akhirnya akan menghasilkan kesimpulan yang salah.

4. *Analisa tanpa mengetahui permasalahan yang sebenarnya.*

5. *Salah memilih satuan pengukuran dari sebuah unjuk kerja.*

6. *Workload yang tidak terwakili:* Pemilihan workload yang digunakan untuk membandingkan dua buah sistem harus mewakili kegunaan sistem

Table 1.4: Rasio Menggunakan Sistem A Sebagai Dasar

Sistem	Workload 1	Workload 2	Rata-rata
A	1	1	1
B	0.5	2	1.25

tersebut di lapangan.

7. *Salah melakukan teknik evaluasi.*

8. *Mengabaikan parameter-parameter penting.*

9. *Mengabaikan faktor-faktor lain yang penting:* Parameter-parameter yang bervariasi disebut sebagai faktor. Misalnya, dari berbagai macam parameter dari workload (e.g., alokasi memori, jumlah pengguna, pola kedatangan permintaan, prioritas dsb) hanya jumlah pengguna yang dipilih sebagai faktor, sedang parameter yang lain dapat dianggap sebagai variable tetap.

10. *Desain uji coba yang tidak sesuai.*

11. *Kompleksitas sistem tidak sesuai.*

12. *Tanpa analisis.*

13. *Melakukan analisa secara sembangan.* Seringkali para analis melakukan beberapa kesalahan dalam hal pengukuran, simulasi dan analisa model karena, misalnya, simulasi yang terlalu pendek.

14. *Tidak ada analisa sensitifitas.*

15. *Mengabaikan kesalahan-kesalahan pada bagian masukan.*

16. *Perlakuan yang tidak tepat terhadap outliers.* Outliers adalah nilai yang terlalu besar atau terlalu kecil dibandingkan dengan nilai mayoritas.

17. *Mengasumsikan tidak ada perubahan di masa depan.*
18. *Mengabaikan keaneka-ragaman.*
19. *Analisa yang terlalu kompleks.*
20. *Presentasi hasil yang tidak tepat.*
21. *mengabaikan aspek-aspek sosial.*
22. *Mengabaikan asumsi dan batasan-batasan.*

Pemilihan Tehnik Evaluasi

Note:

- Secara umum terdapat tiga macam tehnik (cara) yang dapat digunakan untuk melakukan evaluasi terhadap unjuk kerja suatu sistem, yaitu: pemodelan (Analytical modelling), simulasi (simulation) dan pengukuran (measurement).
- Pertimbangan-pertimbangan akan tehnik mana yang paling sesuai diterapkan dalam sebuah proses evaluasi ditunjukkan dalam Tabel 1.5 dari yang paling penting menuju ke hal yang paling tidak penting.
- Kunci utama dalam menentukan tehnik evaluasi adalah *life-cycle stage* dari sistem. Metoda pengukuran hanya dimungkinan apabila sistem yang sama/mirip dengan sistem yang sedang dievaluasi sudah ada. Tetapi apabila sistem ini adalah sesuatu yang baru, maka analytical modeling dan simulation adalah metoda yang paling sesuai digunakan.
- Analytical modeling dan simulation dapat digunakan pada situasi dimana pengukuran tidak dapat dilakukan. Tetapi kedua metoda ini akan lebih berguna apabila dilakukan berdasarkan hasil pengukuran sebelumnya.

Table 1.5: Kriteria Pemilihan Tehnik Evaluasi

Criterion	Analytical Modelling	Simulation	Measurement
1. Life-cycle stage	Any	Any	Postprototype
2. Time required	Small	Medium	Varies
3. Tools	Analyst	Computer languages	Instrumentation
4. Accuracy	Low	Moderate	Varies
5. Trade-off evaluation	Easy	Moderate	Difficult
6. Cost	Small	Medium	High
7. Saleability	Low	Medium	High

- Kriteria berikutnya adalah ketersediaan waktu. Jika dibutuhkan hasil dalam waktu cepat maka analytical modelling adalah metoda yang paling tepat. Metoda simulation membutuhkan waktu cukup lama. Sedangkan measurement membutuhkan waktu bervariasi.

- Kriteria ketiga adalah ketersediaan tool, antara lain: kemampuan melakukan pemodelan, ketrampilan menggunakan bahasa simulasi dan peralatan-peralatan untuk melakukan pengukuran.

- Derajad keakuratan adalah kriteria selanjutnya. Analytical modelling memiliki derajad keakuratan yang paling rendah karena terlalu banyak adanya penyederhanaan dan asumsi. Simulation memiliki derajad keakuratan yang lebih baik dari analytical modelling. Sementara itu derajad keakuratan measurement sangat dipengaruhi oleh hal-hal ekternal, misalnya konfigurasi sistem, jenis dari workload yang digunakan, waktu pengukuran, kepresisian peralatan dsb.

- Tujuan dari evaluasi untuk kerja adalah membandingkan dari beberapa alternatif yang ada atau mencari paramater yang paling optimal. Ana-

lytical modelling secara umum merupakan tehnik terbaik untuk memahami pengaruh dari perubahan-perubahan paramater dan interaksinya (trade-off evaluation).

- Biaya yang dialokasikan untuk evaluasi unjuk kerja sebuah sistem sangat tinggi apabila metoda measurement dipilih. Hal ini disebabkan adanya kebutuhan akan ketersediaan peralatan dan waktu. Sedangkan analytical modelling hanya membutuhkan kertas dan pensil.

- Daya jual (saleability) sebuah sistem akan lebih tinggi apabila metoda measurement dipilih. Untuk hal ini metoda measurement lebih meyakinkan dibandingkan dengan dua metoda yang lain.

- Akan tetapi, untuk hal-hal tertentu, menggunakan kombinasi dari dua atau ketiga metoda yang ada mungkin lebih menguntungkan.

- Do not trust the results of a simulation model until they have been validated by analytical modelling or measurements.

- Do not trust the results of an analytical modelling until they have been validated by a simulation model or measurement.

- Do not trust the results of a measurement until they have been validated by simulation or analytical modelling.

Pemilihan Satuan Ukuran Unjuk Kerja *Note:*

- Setiap layanan yang diminta dari sebuah sistem terdapat beberapa kemungkinan hasil keluaran yang dapat diberikan oleh sistem tersebut, yaitu: sistem memberikan layanan dengan benar, sistem memberikan layanan dengan tidak benar atau sistem menolak memberikan layanan. Perhatikan Gambar 1.1.

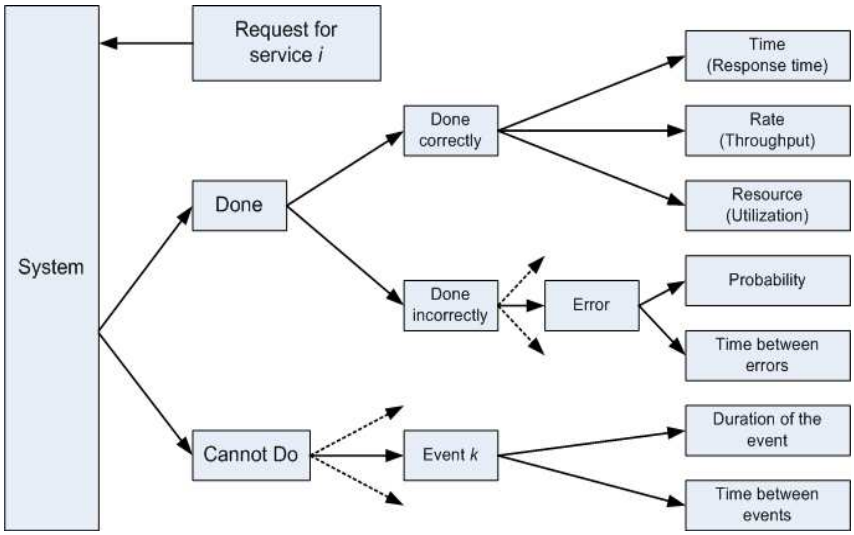

Figure 1.1: Satuan ukuran unjuk kerja.

Satuan Ukuran Unjuk Kerja yang Umum Digunakan *Note:*

- **Response Time** didefinisikan sebagai interval waktu antara permintaan layanan dan respon dari system, seperti terlihat dalam Gambar 1.2.
- **Reaction Time** adalah waktu antara permintaan layanan dan awal dari proses eksekusi.
- **Throughput** didefinisikan sebagai perbandingan antara jumlah permintaan terhadap waktu dari sebuah sistem. Misalnya, untuk CPU

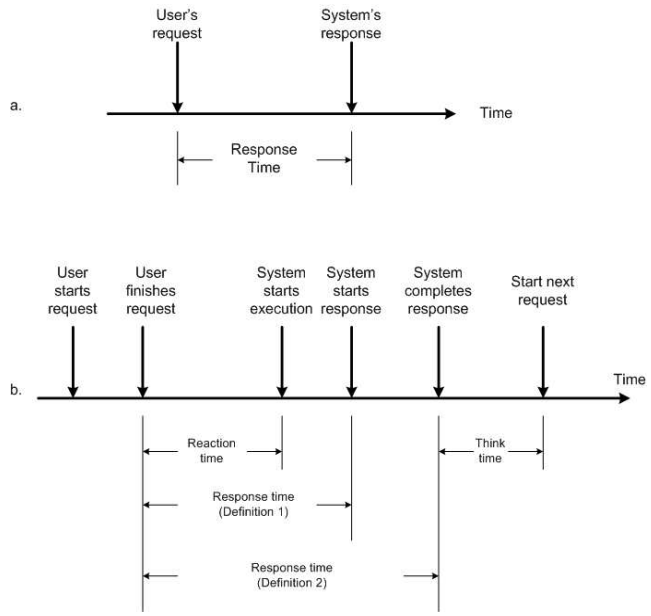

Figure 1.2: Definisi response time.

throughput diukur dengan Millions of instructions per second (MIPS), untuk jaringan throughput dikur dengan jumlah packet per second (pps) atau bit per second (bps). Pada umumnya throughput akan meningkat bersamaan dengan meningkatnya beban bagi sebuah sistem sampai pada titik tertentu, selanjutnya dengan beban tertentu throughput akan mencapai titik steady-state atau bahkan menurun. Throughput maksimum yang dapat dicapai oleh sebuah sistem dengan beban tertentu disebut sebagai kapasitas nominal (*nominal capacity*) dari sebuah sistem. Sebagai contoh dalam teknologi jaringan, yang dimaksud dengan kapasitas nominal adalah *bandwidth*.

- Ratio antara throughput maksimum yang dapat dicapai dengan kapasitas nominal disebut sebagai **efficiency**.

- **Utilization** diukur sebagai sejumlah waktu dimana sebuah sistem sibuk melayani permintaan. Atau dengan kata lain utilization didefinisikan sebagai rasio antara waktu sibuk dan total waktu dalam periode tertentu. Sedangkan periode waktu dimana sistem tidak melakukan kegiatan apapun disebut *idle time*.

- **Reliability** dari sebuah sistem biasanya diukur berdasarkan probabilitas kesalahan atau waktu rata-rata antar kesalahan (mean time between error).

- **Availability** dari sebuah sistem didefinisikan sebagai sejumlah waktu yang dapat disediakan oleh sistem untuk melayani permintaan. Waktu di mana sebuah sistem tidak dapat melayani permintaan disebut sebagai *downtime*. Sedangkan waktu di mana sebuah sistem dapat melayani permintaan disebut sebagai *uptime*. Waktu rata-rata bagi uptime seringkali disebut juga sebagai *Mean Time to Failure (MTTF).*

- **Biaya (cost)** seringkali juga dipakai untuk membandingkan antara dua buah sistem baik hardware maupun software.

Klasifikasi Utilitas dari Satuan Ukuran *Note:*

Berdasarkan fungsi utilitas dari satuan ukuran, dapat dibedakan ke dalam tiga macam, yaitu:

- *Higher is Better (HB).* Dalam hal ini user menyukai performance dengan satuan ukuran yang tinggi. Misalnya, throughput, semakin tinggi throughput semakin disukai.

- *Lower is Better (LB).* Dalam hal ini user menyukai performance dengan satuan ukuran yang rendah. Misalnya, jika sebuah sistem dapat memberikan response time yang kecil, maka sistem tersebut lebih baik daripada sistem yang lain.

- *Nominal is Best (NB).* Nilai satuan ukuran yang terlalu atau terlalu rendah tidak dinginkan. Misalnya, utilisasi. Semakin tinggi utilisasi dari sebuah sistem dianggap jelek, karena ini berarti bahwa sistem tersebut memiliki response time yang tinggi. Utilisasi yang sangat rendah juga tidak diharapkan karena terlalu banyak resource dari sistem yang tidak

terpakai (idle). Biasanya utilisasi yang dianggap baik berada di antara 50 sampai 70 persen.

Figure 1.3: Tipe-tipe satuan ukuran.

Chapter 2

Pengantar MATLAB

Perintah-Perintah Dasar

Note:

MATLAB akan memberikan respons secara langsung terhadap ekspresi apapun yang diketikkan pada editor MATLAB. Sebagai contoh:

```
>> 3 + 2

ans =
     5


>> 2^5

ans =
     32


>> sin(pi/2)

ans =
     1


>> exp(1*pi)
```

```
ans =
     -1.0000 + 0.00001
```

Penugasan nilai ke dalam variabel dapat dilakukan seperti contoh di bawah ini:

```
>> x = sqrt(3)

x =
     1.7321


>> atan(x)

ans =
     1.0472


>> pi/ans

ans =
     3


>> y = exp(log(10))-10

y =
     1.7764e-15
```

m-File

Note:

m-file merupakan editor MATLAB yang berguna sebagai tempat menuliskan script dari kode-kode MATLAB. File ini disimpan dalam bentuk file dengan

Table 2.1: Fungsi matematika dalam MATLAB

Simbol	Operasi
`pi`	π
`exp(n)`	Bilangan natural berpangkat-n, e^n
`log`	Logaritma natural
`log2`	Logaritma dengan basis-2
`log10`	Logaritma dengan basis-10 (desimal)
`sin`	Sinus
`cos`	Cosinus
`tan`	Tangent
`asin`	Inverse sinus
`acos`	Inverse cosinus
`atan`	Inverse tangent

ektensi .m. Script yang tersimpan dalam bentuk m-file dapat dieksekusi secara langsung melalui MATLAB command window.

Vektor

Note:

Membuat vektor dengan MATLAB, sebagai contoh:

```
>> a = [1 2 3 4 5 6 7]

a =

     1 2 3 4 5 6 7
```

Membuat vektor dengan bilangan ascending dengan kenaikan 2:

```
>> t = 0:2:20

t =

     0 2 4 6 8 10 12 14 16 18 20
```

Manipulasi vektor dapat dilakukan seperti contoh di bawah ini:

```
>> b = a + 5

y =
     6 7 8 9 10 11 12


>> c = a + b

c =
     7 9 11 13 15 17 19


>> x = [1 2 3];
>> y = [3 4 5];
>> z = x.*y

z =
     3 8 15
```

Operator dalam MATLAB didefinisikan dalam Tabel 2.2

Matrik (Array)

Note:

Membuat array dengan MATLAB, sebagai contoh:

```
>> B = [1 2 3 4; 5 6 7 8; 9 10 11 12]

B =
```

$$\begin{matrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{matrix}$$

```
>> size(B)
```

Table 2.2: Operator dalam MATLAB

Simbol	Operasi
$*$	Perkalian
/	Pembagian
$+$	Penjumlahan
$-$	Pengurangan
^	Pangkat
.$*$	Perkalian setiap elemen di dalam array
./	Pembagian setiap elemen di dalam array
.^	Pemangkatan setiap elemen di dalam array

```
ans =
     3   3


>> C = ones(1,3)

ans =
     1 1 1


>> C'

ans =
     1
     1
     1
```

Perintah-perintah yang dapat digunakan untuk membangun matriks dalam MATLAB didefinisikan dalam Tabel 2.3

Table 2.3: Perintah membangun matrik

Perintah	Fungsi
`eye`	matrik identitas
`zeros`	matrik dengan semua elemen bilangan 0
`ones`	matrik dengan semua elemen bilangan 1
`diag`	diagonal matrik
`triu`	matrik upper-triangular
`tril`	matrik lower-triangular
`repmat`	duplikasi matrik
`inv`	inverse matrik
`det`	determinan matrik
`sum`	menjumlah setiap elemen kolom dari matrik
`eig`	eigenvalue dan eigenvektor dari matrik

Operasi Matrik

Note:

Operasi perkalian matreik mengikuti aturan perkalian sebuah matrik. Sebagai contoh:

```
>> c = [ 1 10 20 30 ];
>> B*c

ans =
???  Error using ==> mtimes
Inner matrix dimensions must agree.
```

Pesan error ini disebabkan oleh dimensi kedua matrik untuk operasi perkalian tidak sesuai, matrik B berdimensi 3×4 sedangkan matrik c berdimensi 1×4. Agar dimensi kedua matrik bersesuaian, operasi transpose harus dilakukan pada matrik c sebagai berikut:

```
>> c = [ 1 10 20 30 ];
```

```
>> B*c'

ans =

     201

     445

     689


>> help sum
>> sum(B)

ans =

     15 18 21 24
```

Control Flow

Note:

MATLAB mengenal dua macam cara untuk melakukan proses looping atau iterasi, yaitu: *for loop* dan *while loop* dan dua macam cara untuk melakukan seleksi, yaitu: *if-else* dan *switch case*.

For loop memungkinkan sekelompok perintah diulang sebanyak suatu jumlah yang tetap. Contoh:

```
>> for j=1:4
j
end

j =

     1

j =

     2

j =

     3

j =
```

```
     4
```

```
>> for j=1:4
v(j)=j;
end

v =
     1 2 3 4
```

Bagaimana hasil dari contoh script di bawah ini?

```
>> clear all
>> B = [[1 2 3]' [3 2 1]' [2 1 3]']
>> for j=2:3,
for i=j:3,
B(i,:) = B(i,:) - B(j-1,:)*B(i,j-1)/B(j-1,j-1);
end
end
```

While loop akan melakukan perulangan (iterasi) secara terus menerus sampai suatu kondisi tertentu dipenuhi. Contoh:

```
>> i=0;
while i<5
disp(i);
i=i+1;
end

     0
     1
     2
     3
```

```
4
```

Bagaimana hasil dari contoh script di bawah ini, jika diketahui sebuah persamaan differential $y' = x - |y|$, $y(0) = 1$ diaproksimasi dengan metoda Euler?

```
>> h = 0.001;
>> x = [0:h:2];
>> y = 0*x;
>> y(1) = 1;
>> i = 1;
>> size(x)
>> max(size(x))
>> while(i<max(size(x)))
y(i+1) = y(i) + h*(x(i)-abs(y(i)));
i = i + 1;
end
>> plot(x,y,'go')
>> plot(x,y)
```

Switch melakukan pemilihan berdasarkan masing-masing case yang telah didefinisikan. Sebagai contoh, ketikkan code berikut ini ke dalam m-file dan lihatlah hasilnya.

```
bilangan=5;
x=rem(bilangan,2);
switch(x)
    case 1
        disp(['bilangan',num2str(bilangan),...
            'adalah bilangan ganjil'])
    case 2
        disp(['bilangan',num2str(bilangan),...
```

```
            'adalah bilangan genap'])
    otherwise
        disp('Bilangan tidak mungkin ada')
end
```

If-Else melakukan pemilihan berdasarkanhasil tes rasional. Sebagai contoh, ketikkan code berikut ini ke dalam m-file dan lihatlah hasilnya:

```
a = 4;
b = 4;
if (a<b)
    j = -1;
else if (a>b)
    j = 2;
else
    j = 3
end
```

Plotting

Note:

Salah satu keunggulan MATLAB dibandingkan dengan bahasa pemrograman lain adalah kemampuannya untuk menghasilkan plotting grafik hasil simulasi dengan tingkat keakuratan yang cukup tinggi. Sebagai contoh, ketikkan code berikut ini ke dalam m-file dan lihatlah hasilnya:

```
h = 1/16;
x = 0:h:1;
y = 0*x;
y(1) = 1;
for i=2:max(size(y)),
     y(i) = y(i-1) + h/y(i-1);
end true = sqrt(2*x+1);
```

```
figure(1);
plot(x,y,'go',x,true)
figure(2);
plot(x,abs(true-y),'mx')
```

Kedua gambar hasil dari plotting di atas dapat digabungkan ke dalam satu buah frame dengan perintah subplot.

```
figure(3);
subplot(1,2,1);
plot(x,y,'go',x,true)
subplot(1,2,2);
plot(x,abs(true-y),'mx')
```

Lanjutkan dengan mengetikkan code MATLAB di bawah ini:

```
clf
h = h/2;
x1 = 0:h:1;
y1 = 0*x1;
y1(1) = 1;

for i=2:max(size(y1)),
     y1(i) = y1(i-1) + h/y1(i-1);
end
true1 = sqrt(2*x1+1);

plot(x1,y1,'go',x1,true1)
plot(x1,abs(true1-y1),'mx')
subplot(1,2,1);
plot(x,abs(true-y),'mx')
subplot(1,2,2);
plot(x1,abs(true1-y1),'mx')
```

```
title('Errors for h=1/32')
xlabel('x');
ylabel('|Error|');
subplot(1,2,1);
xlabel('x');
ylabel('|Error|');

title('Errors forh=1/16')
```

Tipe garis, simbol dan warna yang dapat digunakan dalam MATLAB ditunjukkan dalam Tabel 2.4

Table 2.4: Tipe garis, simbol dan warna

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dash-dot
c	cyan	+	plus	–	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
b	black	d	diamond		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

Function

Note:

Sebuah *function* ditulis dalam m-file dengan nama file sama dengan nama

function. Sebagai contoh:

```
function [x]=jumlah(a,b)
% Fungsi ini melakukan penjumlahan a dan b
x=a+b;
```

Pada MATLAB window command panggil fungsi tersebut:

```
>> help jumlah
Fungsi ini melakukan penjumlahan a dan b
>> a=5;
>> b=7;
>> hasil = jumlah(a,b)

hasil =
     12
```

LATIHAN

1. Jika diketahui konversi dari Fahrenheit ke Celcius adalah $C = (F - 32) * 5/9$, tentukan nilai celcius untuk suhu 30^oF sampai 200^oF! Kemudian lakukan plotting untuk kedua buah grafik Celcius dan Fahrenheit!

2. Sekelompok mahasiswa memiliki nilai angka sebagai berikut: 35, 56, 78, 97, 67, 45, 85, 77, 62, 40. Konversikan nilai angka tersebut ke nilai huruf jika diketahui: $E < 50$, $50 < D \leq 60$, $60 < C \leq 70$, $70 < B \leq 80$ dan $A > 80$.

Chapter 3

Workload

Pengertian Workload

Note:

- Workload diterjemahkan dalam bahasa Indonesia sebagai 'beban kerja' adalah beban yang diberikan pada sebuah sistem komputer dengan tujuan untuk membandingkan unjuk kerja antara sistem satud engan sistem lainnya.
- Beban kerja dapat berupa *real* atau *synthetic*.
- *Real workload* dilakukan dengan cara melakukan observasi langsung terhadap sistem yang sedang beroperasi. Dengan demikian proses ini tidak dapat diulang, dengan demikian model semacam ini tidak sesuai untuk proses test workload.
- Sedangkan synthetic workload yang memiliki karakteristik mirip dengan real workload dapat dipakai berulang-ulang dan dapat digunakan dipakai sebagai sarana untuk melakukan studi terhadap sebuah sistem.
- Keuntungan utama menggunakan synthetic workload adalah bahwa workload ini merupakan representasi atau model dari real workload.
- Keuntungan lain menggunakan synthetic workload adalah tidak digunakannya real data yang mungkin berukuran besar dan sangat sensitif;

workload juga dapat diubah-ubah sesuai tanpa mengganggu operasi yang lain; workload dapat digunakan/diterapkan untuk sistem yang lain tanpa perubahan yang berarti.

Benchmark Populer

Note:

Istilah benchmark sepada dengan istilah worlkload dan digunakan secara bergantian. Misalnya program-program komputer yang digunakan untuk menguji unjuk kerja sistem (synthetic workload) seringkali disebut sebagai benchmark. Berikut ini adalah bermacam-macam benchmark populer yang sering digunakan.

Sieve

Note:

Kernel Sieve telah digunakan untuk menguji unjuk kerja dari microprocessor, personal computer ataupun bahasa pemgrograman tingkat tinggi. Kernel ini menggunakan algoritma Sieve Eratosthenes dengan cara mencari semua bilangan prima dibawah n. Dalam bentuk bahasa C, bentuk kernel sieve adalah seperti di bawah ini:

```
/*
 * Filename:
 *
 *   sieve.c
 *
 * Description:
 *
 *   The Sieve of Eratosthenes benchmark, from Byte Magazine
 *   early 1980s, when a PC would do well to run this in 10
 *   seconds. This version really does count prime numbers
 *   but omits the numbers 1, 3 and all even numbers. The
 *   expected count is 1899.
```

```
 *
 */

#include <time.h> #include <report.h>
#define SIZE 8190

int sieve () {
  unsigned char flags [SIZE + 1];
  int iter;
  int count;

  for (iter = 1; iter <= 10; iter++)
    {
      int i, prime, k;

      count = 0;

      for (i = 0; i <= SIZE; i++)
        flags [i] = 1;

      for (i = 0; i <= SIZE; i++)
        {
          if (flags [i])
            {
              prime = i + i + 3;
              k = i + prime;

              while (k <= SIZE)
                {
                  flags [k] = 0;
```

```
                k += prime;
              }

            count++;
          }
        }
    }

  return count;
}

int main () {
  int ans;
  clock_t t1, t2;

  test ("sieve", "Sieve benchmark, $Revision: 1.1 $");

  t1 = clock ();
  ans = sieve ();
  t2 = clock ();

  if (ans != 1899)
    failed ("Sieve result wrong, ans = %d, expected 1899", ans);

  comment ("Time taken = %.3e mSecs",
    1000.0 * ((double)t2 - (double)t1) / (double)CLOCKS_PER_SEC);

  result ();
}
```

Ackermann's Function *Note:*

Ackermann benchmark digunakan untuk menguji unjuk kerja sistem dalam melakukan pemanggilan prosedur atau fungsi. Ackerman $(3, n)$ melakukan evaluasi untuk nilai n dari 1 sampai 6. Beberapa parameter yang digunakan untuk membandingkan beberapa sistem adalah: *the average execution time per call*, *the number of instructions executed per call* dan *the amount of stack space required for each call.*

- Nilai dari fungsi Ackermann $(3, n)$ adalah $2^{n+3}-3$. Nilai ini dipakai untuk melakukan verifikasi terhadap implementasi dari ackerman benchmark.
- Jumlah panggilan rekursif dalam melakukan evaluasi Ackermann $(3, n)$ adalah:

  $$(512 \times 4^{n-1} - 15 \times 2^{n+3} + 9n + 37)/3$$

  Ekspresi ini digunakan untuk mengetahui *the execution time per call.*
- Kedalaman maksimum dari pemanggilan prosedur adalah $2^{n+3} - 4$. Jadi jumlah stack space yang dibutuhkan meningkat dua kali pada saatt n ditingkatkan sebanyak 1.

Fungsi Ackermann dalam bahasa C adalah sebagai berikut:

```
/*
 * Filename:
 *
 *   ackermann.c
 *
 * Description:
 *
 *   Ackermann's function "is an example of a recursive function which
 *   is not primitive recursive". It is interesting from the point of
 *   view of benchmarking because it "grows faster than any primitive
 *   recursive function" and gives us a lot of nested function calls
```

```
 *   for little effort.
 *
 *   It is defined as follows:
 *   A(0, n) = n+1
 *   A(m, 0) = A(m-1, 1)
 *   A(m, n) = A(m-1, A(m, n-1))
 *
 *   We use A(3,6) as the benchmark. This used to take long enough to
 *   confirm the execution time with a stopwatch. Nowadays that's out
 *   of the question. BTW, the value of A(4,2) has 19729 digits!
 *
 *   A (3,6) gives us 172233 calls, with a nesting depth of 511.
 *
 * Credits:
 *
 *   Ackermann's function is named for Wilhelm Ackermann, a
 *   mathematical logician who worked Germany during the first half
 *   if the 20th century.
 *
 */

#include <time.h> #include <report.h>


static int A (int m, int n) {
  if (m == 0)
    return n + 1;
  else if (n == 0)
    return A (m - 1, 1);
  else
```

```
    return A (m - 1, A (m, n - 1));
}

int main () {
  int ans;
  clock_t t1, t2;

  test ("ackermann", "Function call benchmark, A (3, 6)");

  t1 = clock ();
  ans = A (3,6);
  t2 = clock ();

  if (ans != 509)
    failed ("Result wrong, got %d, expected %d", ans, 509);

  comment ("time taken = %.3e Seconds",
    ((double)t2 - (double)t1) / (double)CLOCKS_PER_SEC);

  result ();
}
```

Whetstone

Note:

Whetstone benchmark melakukan evaluasi terhadap bermacam-macam kemampuan prosesor seperti: pengalamatan array, aritmetika fixed dan floating point, pemanggilan subrutin dan passing parameter. Hasil dari Whetstone bechmark diukur dalam KWIPS (Kilo Whetstone Instruction Per Second). Misalnya untuk personal komputer modern dapat mencapai nilai 100.000 KWIPS, sedangkan microcontroller Motorola 68020 dapat mencapai 2000 KWIPS pada kecepatan 25MHz..

Whetstone benchmark dalam bahasa C adalah sebagai berikut:

```
/*****************************************************************************
 *
 * Filename:
 *
 *   whetstone.c
 *
 * Description:
 *
 *   Performs one million Whetstone instructions, a number of times, then
 *   prints the execution speed in K Whetstone Instructions per Second
 *   (KWIPS). For example, if ntimes = 1, and the execution time is 1
 *   second, then the result is 1000 KWIPS.
 *
 * Credits:
 *
 *   This source text was translated from the Ada PIWG test A000093 by
 *   Chris Nettleton, November 1996.
 *
 * Revision:
 *
 *   $Id: whetstone.c,v 1.1.1.1 2004/05/10 20:33:29 cvs Exp $
 *
 *****************************************************************************/

#include <report.h> #include <time.h>

/* The KWIPS rating is the number of Kilo Whetstone Instructions
Per
 * Second. The following code is worth 1 million whetstones for each
```

```
 * iteration, so to compute the rating, run the code ntimes, then: rating
 * = 1000 / time taken in secs / ntimes ntimes should be set so the
 * benchmark takes about ten seconds to run so you can get an approximate
 * verification of the printed rating.
 *
 * A 25MHz Motorola 68020 with 68881 scores about 2000 KWIPS, so by setting
 * ntimes to 20, the execution time is 10 secs, which is a good time from
 * the point of view of timing the clock function, and from checking using
 * a stop watch. ntimes = 100 is good on a Pentium.  */

static const int ntimes = 1;

/* Benchmark timing stuff */

static long rating; static clock_t start_time; static clock_t
stop_time;



/*
 * My floating abs (we don't have a math library)
 */
double fabs (double x);



/* These routines are provided for use by ACM SIGAda PIWG for
making
 * measurements. These routines are copyrighted and may only be used for
 * performance measurements. These math routines must not be distributed
 * without this notice. No permission is granted to any party to modify,
 * to redistribute, to sell, to give away, or to otherwise use or transmit
```

```
 * these math routines without express written permission from Westinghous
 * Electric Corporation, c/o Jon Squire, P.O. Box 746 MS1615, Baltimore,
 * MD 21203.  */

const float pi_2 = 1.57079632679489661;

static float sin (float x) {
  const float c1 = 1.57079631847;
  const float c3 = -0.64596371106;
  const float c5 = 0.07968967928;
  const float c7 = -0.00467376557;
  const float c9 = 0.00015148419;

  float x_norm;
  float x_int;
  float x_2;
  float y;

  x_norm = x / pi_2;
  if (fabs (x_norm) > 4.0)
    {
      x_int = (float) ((int) (x_norm / 4.0));
      x_norm = x_norm - 4.0 * x_int;
    }

  if (x_norm > 2.0)
    x_norm = 2.0 - x_norm;
  else if (x_norm < -2.0)
    x_norm = -2.0 - x_norm;
```

```
  if (x_norm > 1.0)
    x_norm = 2.0 - x_norm;
  else if (x_norm < -1.0)
    x_norm = -2.0 - x_norm;

  x_2 = x_norm * x_norm;
  y = (c1 + (c3 + (c5 + (c7 + c9 * x_2) * x_2) * x_2) * x_2) * x_norm;

  return y;
}

static float cos (float x) {
  return sin (x + pi_2);
}

static float atan (float x) {
  const float c1 = 0.9999993329;
  const float c3 = -0.3332985605;
  const float c5 = 0.1994653599;
  const float c7 = -0.1390853351;
  const float c9 = 0.0964200441;
  const float c11 = -0.0559098861;
  const float c13 = 0.0218612288;
  const float c15 = -0.0040540580;

  float a_2;
  float y;
  float a;

  a = x;
```

```
  if (fabs (a) > 1.0)
    a = 1.0 / a;

  a_2 = a * a;
  y =
    (c1 +
     (c3 +
      (c5 +
       (c7 +
    (c9 +
     (c11 +
      (c13 + c15 * a_2) * a_2) * a_2) * a_2) * a_2) * a_2) * a_2) * a_2) * a;

  if (fabs (x) >= 1.0)
    {
      if (x < 0.0)
    y = -(pi_2 + y);
      else
    y = pi_2 - y;
    }

  return y;
}

static float sqrt (float x) {
  float y, root_pwr, x_norm;
  const float a = 2.1902;
  const float b = -3.0339;
  const float c = 1.5451;
```

```
  x_norm = x;
  root_pwr = 1.0;

  if (x <= 0.0)
    return 0.0;

  if (x > 1.0)
    {
      while (x_norm > 1.0)
    {
      root_pwr = root_pwr * 2.0;
      x_norm = x_norm * 0.25;
    }
    }
  else
    {
      while (x_norm < 0.25)
    {
      root_pwr = root_pwr * 0.5;
      x_norm = x_norm * 4.0;
    }
    }

  y = a + b / (c + x_norm);
  y = 0.5 * (y + x_norm / y);
  y = 0.5 * (y + x_norm / y);
  y = y * root_pwr;

  return y;
}
```

```
static float exp (float x) {
  const float c1 = 9.99999900943303e-01;
  const float c2 = 5.00006347344554e-01;
  const float c3 = 1.66667985598315e-01;
  const float c4 = 4.16350120350139e-02;
  const float c5 = 8.32859610677671e-03;
  const float c6 = 1.43927433449119e-03;
  const float c7 = 2.04699933614437e-04;

  float x1;
  float y;
  float e_pwr = 1.0;
  float e = 2.71828182845905;

  x1 = fabs (x);
  if (x1 > 88.0)
    return 0.0;

  while (x1 >= 1.0)
    {
      e_pwr = e_pwr * e * e;
      x1 = x1 - 2.0;
    }

  y =
    1.0 + (c1 +
       (c2 +
        (c3 +
         (c4 + (c5 + (c6 + c7 * x1) * x1) * x1) * x1) * x1) * x1) * x1;
  y = y * e_pwr;
```

```
  if (x < 0.0)
    y = 1.0 / y;

  return y;
}

static float log10 (float x) {
  const float c1 = 0.868591718;
  const float c3 = 0.289335524;
  const float c5 = 0.177522071;
  const float c7 = 0.094376476;
  const float c9 = 0.191337714;
  const float c_r10 = 3.1622777;

  float y;
  float x_norm;
  float x_log;
  float frac;
  float frac_2;

  x_log = 0.5;
  x_norm = x;
  if (x <= 0.0)
    return 0.0;

  if (x >= 10.0)
    {
      while (x_norm >= 10.0)    /* reduce to 1.0 .. 10.0 */
    {
      x_log = x_log + 1.0;
```

```
      x_norm = x_norm * 0.1;
    }
    }
  else
    {
      while (x_norm < 1.0)  /* reduce to 1.0 .. 10.0 */
    {
      x_log = x_log - 1.0;
      x_norm = x_norm * 10.0;
    }
    }

  frac = (x_norm - c_r10) / (x_norm + c_r10);
  frac_2 = frac * frac;
  y = (c1 + (c3 + (c5 + (c7 + c9 * frac_2) * frac_2) * frac_2) * frac_2)
    * frac;

  return y + x_log;
}


static float log (float x) {
  return 2.302585093 * log10 (x);
}

/*
 * End of copyrighted section
 */
```

```
float x1, x2, x3, x4, x, y, z, t, t1, t2; float e1[5]; int j, k,
l, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11;


static void pa (float *e) {
  /* tests computations with an array as a parameter */
  int j;
  /* T,T2 : FLOAT are global variables */

  j = 0;
LAB:
  e[1] = (e[1] + e[2] + e[3] - e[4]) * t;
  e[2] = (e[1] + e[2] - e[3] + e[4]) * t;
  e[3] = (e[1] - e[2] + e[3] + e[4]) * t;
  e[4] = (-e[1] + e[2] + e[3] + e[4]) / t2;

  j = j + 1;
  if (j < 6)
    goto LAB;
}

static void p0 () {
  /*
   * tests computations with no parameters
   * T1,T2 : FLOAT are global
   * E1 : VECTOR (1..4) is global
   * J,K,L : INTEGER are global
   */

  e1[j] = e1[k];
```

```
  e1[k] = e1[l];
  e1[l] = e1[j];
}

static void p3 (float *x, float *y, float *z) {
  /*
   * tests computations with simple identifiers as parameters
   * T,T2 : FLOAT are global
   */

  *x = t * (*x + *y);
  *y = t * (*x + *y);
  *z = (*x + *y) / t2;
}

/*
 * Whetstone proper starts here
 */
int main () {
  int I = 10;           /* loop count weighting factor */
  int cycle_no;         /* major loop counter */
  int i;            /* loop counter */

  test ("whetstone", "Whetstone scientific benchmark");

  /* Set constants */
  t = 0.499975;
  t1 = 0.50025;
  t2 = 2.0;
```

```
  /* Compute the execution frequency for the benchmark modules */
  n1 = 0;           /* Module 1 not executed */
  n2 = 12 * I;
  n3 = 14 * I;
  n4 = 345 * I;
  n5 = 0;           /* Module 5 not executed */
  n6 = 210 * I;
  n7 = 32 * I;
  n8 = 899 * I;
  n9 = 616 * I;
  n10 = 0;          /* Module 10 not executed */
  n11 = 93 * I;

  start_time = clock ();    /* Get Whetstone start time */

cycle_loop:
  for (cycle_no = 1; cycle_no <= ntimes; cycle_no++)
    {
      /* Module 1 : computations with simple identifiers */
      x1 = 1.0;
      x2 = -1.0;
      x3 = -1.0;
      x4 = -1.0;
      for (i = 1; i <= n1; i++)
    {
      x1 = (x1 + x2 + x3 - x4) * t;
      x2 = (x1 + x2 - x3 + x4) * t;
      x3 = (x1 + x2 + x3 + x4) * t;
      x4 = (-x1 + x2 + x3 + x4) * t;
    }
```

```
  /* end Module 1 */

  /* Module 2: computations with array elements */
  e1[1] = 1.0;
  e1[2] = -1.0;
  e1[3] = -1.0;
  e1[4] = -1.0;
  for (i = 1; i <= n2; i++)
{
  e1[1] = (e1[1] + e1[2] + e1[3] - e1[4]) * t;
  e1[2] = (e1[1] + e1[2] - e1[3] + e1[4]) * t;
  e1[3] = (e1[1] - e1[2] + e1[3] + e1[4]) * t;
  e1[4] = (-e1[1] + e1[2] + e1[3] + e1[4]) * t;
}
  /* end Module 2 */

  /* Module 3 : passing an array as a parmeter */
  for (i = 1; i <= n3; i++)
pa (e1);
  /* end Module 3 */

  /* Module 4 : performing conditional jumps */
  j = 1;
  for (i = 1; i <= n4; i++)
{
  if (j == 1)
    j = 2;
  else
    j = 3;
  if (j > 2)
```

```
      j = 0;
    else
      j = 1;
    if (j < 1)
      j = 1;
    else
      j = 0;
  }
    /* end Module 4 */

    /* Module 5 : omitted */

    /* Module 6 : performing integer arithmetic */
    j = 1;
    k = 2;
    l = 3;
    for (i = 1; i <= n6; i++)
  {
    j = j * (k - j) * (l - k);
    k = l * k - (l - j) * k;
    l = (l - k) * (k + j);
    e1[l - 1] = (float) (j + k + l);
    e1[k - 1] = (float) (j * k * l);
  }
    /* end Module 6 */

    /* Module 7 : performing computations using trigonometric
       functions */
    x = 0.5;
    y = 0.5;
```

```
  for (i = 1; i <= n7; i++)
{
  x =
    t * atan (t2 * sin (x) * cos (x) /
          (cos (x + y) + cos (x - y) - 1.0));
  y =
    t * atan (t2 * sin (y) * cos (y) /
          (cos (x + y) + cos (x - y) - 1.0));
}
  /* end Module 7 */

  /* Module 8 : procedure calls with simple identifiers as
     parameters */
  x = 1.0;
  y = 1.0;
  z = 1.0;
  for (i = 1; i <= n8; i++)
p3 (&x, &y, &z);
  /* end Module 8 */

  /* Module 9 : array reference and procedure calls with no
     parameters */
  j = 1;
  k = 2;
  l = 3;
  e1[1] = 1.0;
  e1[2] = 2.0;
  e1[3] = 3.0;
  for (i = 1; i <= n9; i++)
p0 ();
```

```
    /* end Module 9 */

    /* Module 10 : integer arithmetic */
    j = 2;
    k = 3;
    for (i = 1; i <= n10; i++)
  {
    j = j + k;
    k = k + j;
    j = k - j;
    k = k - j - j;
  }
    /* end Module 10 */

    /* Module 11 : performing computations using standard
       mathematical functions */
    x = 0.75;
    for (i = 1; i <= n11; i++)
  x = sqrt (exp (log (x) / t1));
    /* end Moudle 11 */

  }

/* Get stop time after ntimes */
stop_time = clock ();

/* Now compute number of K Whetstones per sec */
rating = (long) (1000.0 /
         (((double) stop_time - (double) start_time) /
          (double) CLOCKS_PER_SEC / (double) (ntimes)));
```

```
  comment ("Rating = %ld KWIPS", rating);

  if (rating < 100.0 || rating > 100000.0)
    failed ("measured rating out of limits");

  result ();
}
```

Karakterisasi Workload

Note:

Agar sebuah test terhadap beberapa alternatif dapat dilakukan pada kondisi yang sama berulang-ulang, maka workload harus dapat diulang juga. Karena kondisi nyata tidak dapat diulang, maka perlu dilakukan studi terhadap sistem real tersebut, pengamatan terhadap karakteristik kunci dan membangun sebuah model workload yang dapat diulang-ulang. Proses ini disebut sebagai *workload characterization.*

Begitu model telah tersedia, efek-efek perubahan terhadap model dan sistem dapat dilakukan dengan melakukan perubahan terhadap parameter2 dari model tersebut.

Karakterisasi workload dapat dilakukan dengan beberapa teknik berikut:

1. Averaging
2. Specifying dispersion
3. Single-parameter histograms
4. Multi-parameter histograms
5. Principal-component analysis
6. Markov models
7. Clustering

Averaging

Salah cara paling mudah untuk mengkarakterisasi parameter dari workload adalah dengan menggunakan averaging (rata-rata). Apabila terdapat data sebanyak n yaitu $x_1, x_2, \ldots, x_n$, arithmetic mean dari data tersebut didapatkan dengan persamaan:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3.1}$$

Akan tetapi arithmetic mean tidak dapat diterapkan pada semua model data. Model-model averang yang lain dapat dilihat pada Appendix A.

Dispersi

Rata-rata tidak dapat mewakili sample yang diambil apabila tingkat variansi dari data sample cukup besar. Karena dibutuhkan parameter lain untuk menghitung variansi dari data yaitu varian, dengan rumusan:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{3.2}$$

Standar deviasi, s, yang merupakan akar dari varian lebih disukai karena memiliki satuan yang dengan mean. Sedangkan perbandingan antara standar deviasi dan mean disebut sebagai *Coefficient of Variation (C.O.V)*.

Apabila nilai dari C.O.V adalah nol, maka sample data adalah konstan. Tetapi apabila nilai dari C.O.V tinggi, berari variansi dari data tersebut cukup besar, dalam hal ini menggunakan mean sebagai parameter tidak berguna sama sekali.

Contoh: beberapa program belajar diuji cobakan pada 6 buah universitas dan dilakukan pengukuran selama 6 bulan. Hasil pengukuran terhadap kinerja server ditunjukkan dalam Tabel 3.1.

Berdasarkan contoh kasus di atas, dapat dilihat bahwa C.O.V untuk Tabel 3.1 terlihat sangat besar, hal ini mengindikasikan bahwa menggabungkan semua program pada satu kelas tidak akan memberikan hasil optimal. Karena itu pelaksanaan program seharusnya dibagi ke dalam beberapa kelas. Setelah dibagi dalam beberapa kelas, hasil pengukuran terhadap kinerja server

Table 3.1: Workload Karakteristik

Data	Average	Coefficient of Variation
CPU time (VAX-11/780)	2.19 seconds	40.23
Number of direct write	8.20	53.59
Direct-write byte	10.21 kbytes	82.41
Number of direct read	22.64	25.65
Direct-read byte	49.70 kbytes	21.01

Table 3.2: Workload Karakteristik Setelah Perbaikan

Data	Average	Coefficient of Variation
CPU time (VAX-11/780)	2.57 seconds	3.54
Number of direct write	19.74	4.33
Direct-write byte	13.46 kbytes	3.87
Number of direct read	37.77	3.73
Direct-read byte	36.93 kbytes	3.16

ditunjukkan dalam Tabel 3.2.

Selain menggunakan varian, pengukuran dispersi dapat juga dilakukan dengan menggunakan quartile. Quartile membagi data ke dalam empat bagian pada 25% (Q_1), 50% Q_2 dan 75% Q_3. Nilai quartile ke-α dapat diestimasi dengan mengurutkan data terlebih dahulu dan mengambil elemen ke $[(n-1)\alpha+1]$. [.] mengindikasikan pembulatan ke nilai integer terdekat. Jarak antara Q_3 dan $Q1$ seringkali disebut sebagai *interquartile range*, sedangkan separuh dari interquartile range disebut sebagai *Semi-Interquartile Range (SIQR)*.

$$SIQR = \frac{Q_3 - Q_1}{2} = \frac{x_{0.75} - x_{0.25}}{2} \tag{3.3}$$

Dispersi juga dapat diukur dengan *mean absolute deviation*, yang didefinisikan dengan

$$Meanabsolutedeviation = \frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}| \tag{3.4}$$

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) merupakan salah satu cara yang dapat digunakan untuk mengklasifikasikan komponen dari workload. Seringkali dalam suatu pengukuran sistem, kita mendapati suatu sistem dapat dikarakterisasikan ke dalam 2 atau bahkan lebih variabel. Karena itu semakin besar jumlah variabel yang berperan dalam pengukuran, maka proses komputasi semakin besar. PCA adalah suatu metode tranformasi linear yang mentransformasi variabel-variabel tersebut ke dalam satu variabel saja, disebut principal factor (selanjutnya variabel hasil transformasi ini yang digunakan untuk proses analisa). PCA merupakan transformasi linier yang melakukan klasifikasi dengan cara mentransformasikan data ke koordinat baru sedemikian rupa sehingga data dengan varian terbesar akan berada di sekitar koordinat baru tersebut.

1. Proses transformasi dilakukan dengan persamaan:

$$y_i = \sum_{j=1}^{n} a_{ij}x_j \tag{3.5}$$

   dimana y_i disebut sebagai *principal factor*. Secara statistik, jika diberikan satu set data yang terdiri atas n parameter $\{x_1, x_2, \ldots x_n\}$, PCA akan menghasilkan satu set factor $\{y_1, y_2, \ldots y_n\}$ sedemikian rupa sehingga y's merupakan kombinasi linier dari x's. Sedangkan a_{ij} disebut sebagai *loading* dari variabel x_j pada factor y_i.

2. factor $y's$ membentuk sebuah set yang orthogonal, yang memiliki inner product sebesar 0 (nol):

$$\langle y_i, y_j \rangle = \sum_{k} a_{ik}a_{kj} = 0 \tag{3.6}$$

   Ini juga berarti bahwa y_i's saling tidak berkorelasi satu dengan yang lain.

3. y's membentuk urutan shingga y_1 memiliki prosentase tertinggi dari varian, y_2 memiliki prosentase lebih rendah dan y_n memiliki prosentase terendah. Biasanya hanya factor dengan urutan teratas yang dapat digunakan untuk klasifikasi workload.

Contoh: Pengujian dilakukan pada beberapa terminal pada sebuah LAN. Jumlah paket yang terkirim disimbolkan sebagai x_s sedangkan jumlah paket yang diterima disimbolkan dengan x_r. Hasil pengukuran ditunjukkan pada Tabel 3.3, sedangkan grafik yang menunjukkan hubungan antara paket terkirim dan paket diterima ditunjukkan pada Gambar 3.1.

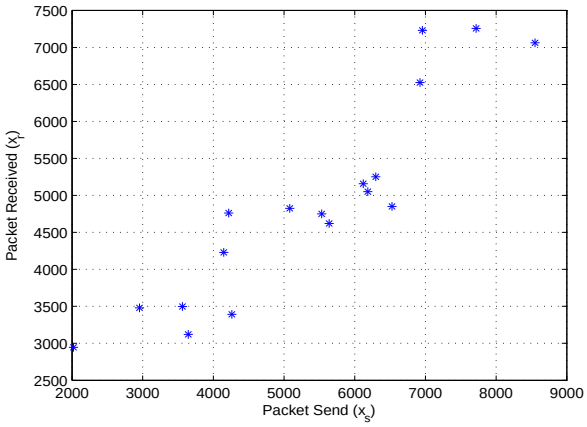

Figure 3.1: Plot packet terkirim dan diterima.

Langkah-langkah penyelesaian dengan PCA:

1. Tentukan mean dan standar deviasi untuk variabel x_s dan x_r. Lihat pada baris terakhir dari Tabel 3.3.

2. Normalisasi setiap variabel ke mean = 0 dan standar deviasi = 1, sebagai berikut:

$$x_s^{'} = \frac{x_s - \bar{x_s}}{s_{x_s}} = \frac{x_s - 5352}{1741}$$
$$x_r^{'} = \frac{x_r - \bar{x_r}}{s_{x_r}} = \frac{x_r - 4889}{1380}$$

Hasil normalisasi ditunjukkan pada Tabel 3.3 pada kolom 4 dan 5.

3. Hitung korelasi antar variabel tersebut:

$$R_{x_s,x_r} = \frac{1}{n-1}\frac{\sum_{i=1}^{n}(x_{si} - \bar{x_s})(x_{ri} - \bar{x_r})}{s_{x_s}s_{x_r}} = 0.916$$

4. Buat matrix korelasi seperti di bawah ini:

$$\mathbf{C} = \begin{bmatrix} 1.000 & 0.916 \\ 0.916 & 1.000 \end{bmatrix}$$

5. Hitung nilai dari eigenvalue dari matrix korelasi tersebut. Eigenvalue dapat dihitung dengan menyelesaikan persamaan karakteristik $|\lambda I - C| = 0$, dimana I merupakan matrix identitas.

$$|\lambda I - C| = \begin{vmatrix} \lambda - 1 & -0.916 \\ -0.916 & \lambda - 1 \end{vmatrix} = 0$$

sehingga didapatkan:

$$(\lambda - 1)^2 - 0.916^2 = 0$$

Nilai eigenvalue adalah 1.916 dan 0.084.

6. Hitung nilai eigenvector ($\mathbf{q}$) dari matriks korelasi, dimana eigenvector $\mathbf{q_1}$ berkaitan dengan eigenvalue $\lambda_1 = 1.916$.

$$\mathbf{C}\mathbf{q_1} = \lambda_1 \mathbf{q_1}$$

atau

$$\begin{bmatrix} 1.000 & 0.916 \\ 0.916 & 1.000 \end{bmatrix} \times \begin{bmatrix} q_{11} \\ q_{21} \end{bmatrix} = 1.916 \begin{bmatrix} q_{11} \\ q_{21} \end{bmatrix}$$

atau

$$q_{11} = q_{21} \tag{3.7}$$

Dengan membatasi bahwa panjang dari eigenvector = 1, maka kita dapatkan eigenvector pertama adalah:

$$\mathbf{q1} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

Dan dengan cara yang sama didapatkan eigenvector kedua:

$$\mathbf{q2} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

7. Dapatkan nilai dari principal factor dengan mengalikan eigenvector dengan variabel ternormalisasi:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{x_s - 5352}{1741} \\ \frac{x_r - 4889}{1380} \end{bmatrix} \tag{3.8}$$

8. Hitung nilai dari principal factor untuk semua observasi. Lihat Tabel 3.3.

9. Hitung jumlah dan jumlah pangkat dua dari principal factor. Dari Tabel terlihat bahwa jumlah pangkat dua dari y_1 dan y_2 adalah 32.565 dan 1.435. Sehingga factor pertama berarti 32.565/(32.565+1.435) atau 95.7% dari variasi. Sedangkan factor kedua hanya 4.3 dari variasi sehingga dapat diabaikan. Dengan demikian nilai-nilai pada faktor pertama tersebut dapat digunakan untuk mengklasifikasikan setiap terminal misalnya terminal dengan load rendah, sedang, atau tinggi.

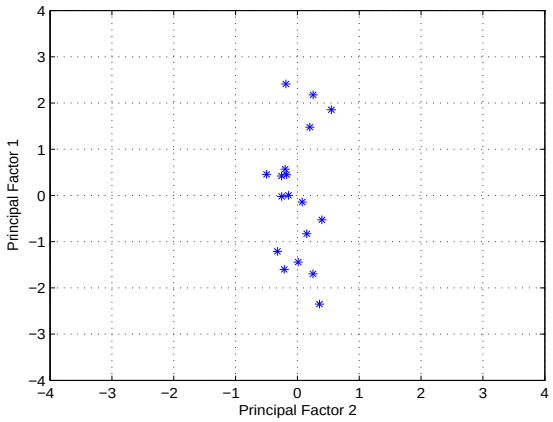

Figure 3.2: Principal factor 1 dan Principal factor 2.

10. Perhatikan hasil plotting dari kedua principal factor seperti terlihat dalam gambar 3.3. Perhatikan bahwa variasi terbesar berada pada Principal

Factor 1. Variasi sepanjang Principal Factor 2 dapat diabaikan. Dengan demikian kita dapat menggunakan Principal Factor 1 untuk melakukan analisa lebih lanjut terhadap data.

Table 3.3: Tabel Analisa dengan PCA

	Variabel		Variabel Ternormalisasi		Principal Factors	
Observasi ke	x_s	x_r	$x_s^{'}$	$x_r^{'}$	y_1	y_2
1	7718	7258	1.359	1.717	2.175	-0.253
2	6958	7232	0.922	1.698	1.853	-0.549
3	8551	7062	1.837	1.575	2.413	0.186
4	6924	6526	0.903	1.186	1.477	-0.200
5	6298	5251	0.543	0.262	0.570	0.199
6	6120	5158	0.441	0.195	0.450	0.174
7	6184	5051	0.478	0.117	0.421	0.255
8	6527	4850	0.675	-0.029	0.457	0.497
9	5081	4825	-0.156	-0.047	-0.143	-0.077
10	4216	4762	-0.652	-0.092	-0.527	-0.396
11	5532	4750	0.103	-0.101	0.002	0.145
12	5638	4620	0.164	-0.195	-0.022	0.254
13	4147	4229	-0.692	-0.479	-0.828	-0.151
14	3562	3497	-1.028	-1.009	-1.441	-0.013
15	2955	3480	-1.377	-1.022	-1.696	-0.251
16	4261	3392	-0.627	-1.085	-1.211	0.324
17	3644	3120	-0.981	-1.283	-1.601	0.213
18	2020	2946	-1.914	-1.409	-2.349	-0.357
$\sum x$	96,336	88,009	0.000	0.000	0.000	0.000
$\sum x^2$	567,119,488	462,661,024	17.000	17.000	32.565	1.435
Mean	5352.0	4889.4	0.000	0.000	0.000	0.000
Standard deviation	1741.0	1379.5	1.000	1.000	1.384	0.290

Clustering

Secara umum hasil pengukuran dari satu atau beberapa workload terdiri atas sejumlah besar komponen. Misalnya, hasil pengukuran dari waktu CPU (CPU time) dapat terdiri atas beberapa puluh, ratusan atau bahkan ribuan data. Karena itu untuk kebutuhan analisa data, akan sangat berguna apabila data-data yang memiliki kemiripan sifat dapat dikelompokkan ke dalam satu cluster tersendiri. Salah satu metoda clustering yang akan dibicarakan dalam bagian ini adalah *Minimum Spanning Tree Method*. Adapun langkah-langkah proses melakukan clustering adalah sebagai berikut:

- Mulai dengan cluster $k = n$.
- Tentukan titik tengah dari cluster ke-i, dimana $i = 1, 2, \cdots, k$. Titik tengah memiliki parameter nilai sama dengan rata-rata semua titik di dalam cluster.
- Hitung matrik jarak antar cluster.
- Tentukan elemen terkecil dan tidak nol dari jarak tersebut. Gabungkan dua buah cluster dengan jarak terpendek.
- Ulangi langkah 2 sampai 4 sampai semua komponen berada dalam satu cluster.

Contoh: Perhatikan 5 buah workload yang diterapkan pada dua buah parameter di bawah ini. CPU Time dan Jumlah Disk I/O diukur dengan 5 buah program. Data setelah diskalakan adalah sebagai berikut:

Langkah 1. Mula-mula anggaplah terdapat cluster sejumlah i sesuai dengan jumlah program.

Langkah 2. Nilai tengah dari masing-masing cluster adalah $\{2, 4\}, \{3, 5\}$, $\{1, 6\}, \{4, 3\}$ dan $\{5, 2\}$.

Langkah 3. Dengan menggunakan rumusan jarak Euclidean, jarak antar matrik adalah sebagai berikut:

Table 3.4: Data untuk proses clustering

Program	CPU Time	Disk I/O
A	2	4
B	3	5
C	1	6
D	4	3
E	5	2

	Program				
Program	A	B	C	D	E
A	0	$\sqrt{2}$	$\sqrt{5}$	$\sqrt{5}$	$\sqrt{13}$
B		0	$\sqrt{5}$	$\sqrt{5}$	$\sqrt{13}$
C			0	$\sqrt{18}$	$\sqrt{32}$
D				0	$\sqrt{2}$
E					0

Rumus Jarak Euclidean untuk dua buah komponen $\{x_{i1}, x_{i2}, \cdots, x_{in}\}$ dan $\{x_{j1}, x_{j2}, \cdots, x_{jn}\}$ adalah:

$$d = \left\{ \sum_{k=1}^{n} (x_{ik} - x_{jk})^2 \right\}^{0.5} \tag{3.9}$$

Langkah 4. Seperti terlihat dalam tabel, jarak terpendek antar cluster adalah $\sqrt{2}$ antara A dan B dan antara D dan E. Karena A dan B dapat digabungkan ke dalam sebuah cluster, demikian pulan D dan E.

Langkah 2. Titik tengah dari cluster pasangan AB adalah $\{(2+3) \div 2, (4+5) \div 2\}$, yaitu $\{2.5, 4.5\}$. Sedangkan titik tengah dari cluster pasangan DE adalah $\{4.5, 2.5\}$.

Langkah 3. Maka sekarang terdapat tiga buah cluster sebagai berikut:

Langkah 4. Seperti terlihat dalam tabel, jarak terpendek antar cluster adalah $\sqrt{4.5}$ antara AB dan C. Karena itu dua buah cluster ini dapat diga-

	Program		
Program	AB	C	DE
AB	0	$\sqrt{4}.5$	$\sqrt{8}$
C		0	$\sqrt{2}4.5$
DE			0

bungkan menjadi sebuah cluster.

Langkah 2. Titik tengah cluster ABC adalah $\{(2+3+1) \div 3, (4+5+6) \div 3\}$ yaitu $\{2, 5\}$.

Langkah 3. Matriks jarak sekarang menjadi:

	Program	
Program	ABC	DE
ABC	0	$\sqrt{1}2.5$
DE		0

Langkah 4. Jarak antar cluster terpendek adalah $\sqrt{1}2.5$. Karena itu kedua cluster tersebut dapat digabungkan menjadi cluster baru ABCDE.

Dendogram untuk proses clustering di atas dapat digambarkan sebagai berikut:

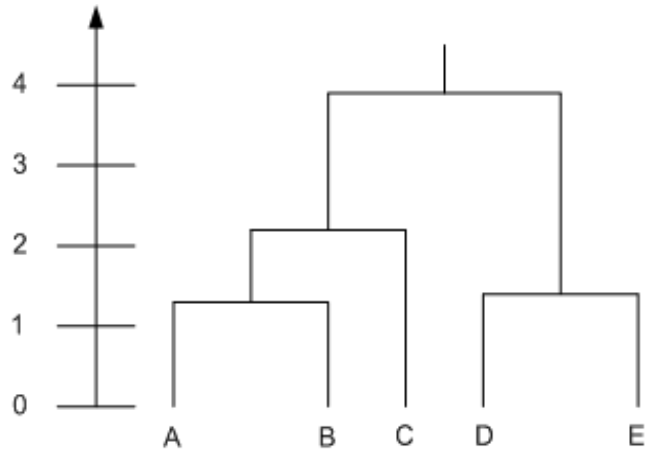

Figure 3.3: Dendogram untuk cluster ABCDE.

Chapter 4

Ratio Games

Rasio merupakan salah satu cara yang dapat digunakan untuk membandingkan unjuk kerja sebuah sistem dengan sistem lainnya. Rasio terdiri atas pembilang (numerator) dan penyebut (denominator). Penyebut seringkali juga disebut sebagai basis (base). Dua buah rasio dengan basis yang berbeda tidak dapat digunakan sebagai pembanding. Akan tetapi terdapat banyak contoh kasus dalam jurnal-jurnal ilmiah dimana proses pembandingan dengan basis yang berbeda dilakukan. Teknik menggunakan rasio untuk dapat memberikan suatu hasil yang berguna disebut sebagai *ratio game*. Bab ini membicarakan teknik dan strategi menggunakan ratio game untuk dapat memberikan hasil yang tepat.

Memilih Basis yang Sesuai *Note:*

Cara termudah menggunakan ratio game adalah dengan cara mempresentasikan unjuk kerja dari dua atau lebih sistem dengan menggunakan berbagai macam workload, mengambil rasio unjuk kerja dari setiap workload dan menggunakan rata-rata untuk menunjukkan bahwa satu sistem lebih baik daripada sistem yang lain.

Sebagai contoh, perhatikan perbandingan unjuk kerja dari 2 buah processor 6502 dan 8080 dalam Tabel 4.1 Unjuk kerja diukur dengan menggunakan dua buah workload, yaitu Block move dan Sieve. Nilai rata-rata dari 10 kali pengukuran untuk setiap sistem ditunjukkan dalam tabel. Analisa terhadap

data dalam total rata-rata, rasio rata-rata dengan sistem A sebagai basis dan ratio rata-rata dengan sistem B sebagai basis ditunjukkan dalam tabel.

Table 4.1: Perbandingan Unjuk Kerja dari 6502 dan 8080

Raw Measurement			With 6502 as a Base		With 8080 as a Base	
	System		System		System	
Bencmark	6502	8080	6502	8080	6502	8080
Block	41.16	51.50	1.00	1.25	0.80	1.0
Sieve	63.17	48.08	1.00	0.76	1.30	1.0
Sum	104.33	99.58	2.00	2.01	2.11	2.0
Average	52.17	49.79	1.00	1.01	1.06	1.0

Berdasarkan tabel di atas didapatkan tiga analisa yang menghasilkan tiga macam konklusi:

- Menggunakan total rasio: unjuk kerja dari 6502 lebih buruk, dimana 6502 menggunakan waktu 4.7% lebih banyak daripada 8080.
- Menggunakan 6502 sebagai basis: unjuk kerja dari 6502 lebih baik, dimana 6502 menggunakan waktu 1% lebih sedikit daripada 8080.
- Menggunakan 8080 sebagai basis: unjuk kerja dari 6502 lebih buruk, dimana 6502 menggunakan waktu 6% lebih banyak daripada 8080.

Menggunakan Rasio yang Sesuai

Note:

Cara lain untuk menunjukkan bahwa unjuk kerja suatu sistem lebih baik dari yang lain adalah dengan cara memilih satuan ukuran unjuk kerja yang sesuai, yaitu rasio antara dua satuan ukuran yang berbeda.

Perhatikan contoh dalam Tabel 4.2 dibawah ini:

Table 4.2: Perbandingan Unjuk Kerja Dua Arsitektur Jaringan

Network	Throughput	Response
A	10	2
B	4	1

Dalam Tabel 4.2 terlihat bahwa dua buah satuan unjuk kerja dari jaringan diukur, yaitu: throughput dan response waktu. Perhatikan bahwa Jaringan A memiliki throughput yang lebih tinggi, tetapi juga memiliki response waktu yang lebih tinggi. Untuk melakukan analisa terhadap kasus di atas, kita gunakan Power sebagai rasio dari throughput dan response waktu. Sperti ditunjukkan dalam Tabel 4.3.

Table 4.3: Perbandingan Unjuk Kerja Dua Arsitektur Jaringan dengan Rasio

Network	Throughput	Response	Power
A	10	2	5
B	4	1	4

Hasil yang sudah ditransformasikan ke dalam Tabel 4.3 memberikan hasil konklusi bahwa Jaringan A lebih baik daripada Jaringan B.

Menggunakan Peningkatan Unjuk Kerja Relatif *Note:*

Dalam kasus-kasus tertentu satuan ukuran telah ditentukan. Untuk mengukur unjuk kerja dari sistem dapat digunakan dengan cara menghitung peningkatan unjuk kerja relatif dari sistem tersebut dengan catatan bahwa workload dicobakan pada dua mesin yang berbeda.

Sebagai contoh, dua buah akselerator floating-point A dan B dicobakan pada dua buah mesin. Satuan pengukuran yang ditentukan adalah MFLOPS (Millions of Floating Point Operation per Seconds) diukur pada dua buah

mesin masing-masing dengan dan tidak dengan akselerator. Perhatikan tabel hasil pengukuran pada Tabel 4.4.

Table 4.4: Perbandingan Unjuk Kerja Dua Akselerator

Alternative	Without	With	Ratio
A on X	2	4	2.00
B on Y	3	5	1.66

Kesimpulan yang dapat ditarik adalah: A memberikan unjuk kerja yang lebih baik dari B.

Ratio Game dengan Prosentase

Note:

Prosentase pada dasarnya adalah rasio. Tetapi seringkali tidak disadari sehingga prosentase dipergunakan tidak pada semestinya. Sebagai contoh perhatikan Tabel 4.5.

Table 4.5: Dua Pengujian pada Dua Sistem

System A				System B			
Test	Total	Pass	% Pass	Test	Total	Pass	% Pass
1	300	60	20	1	32	8	25
2	50	2	4	2	500	40	8
	350	62	20.6		532	48	9

Tabel di atas dapat dianalisa dengan menggunakan 2 buah alternatif. Alternatif pertama dilakukan dengan cara membandingkan masing-masing percobaan secara individu untuk kedua sistem. Perhatikan Gambar 4.1 Dari Gambar tersebut terlihat bahwa System B lebih baik daripada sistem A. Tetapi jika alternatif kedua digunakan dengan cara membandingkan hasil penjumlahan dari kedua buah pengujian, lihat Gambar 4.1, maka didapati bahwa sistem A

lebih baik daripada sistem B.

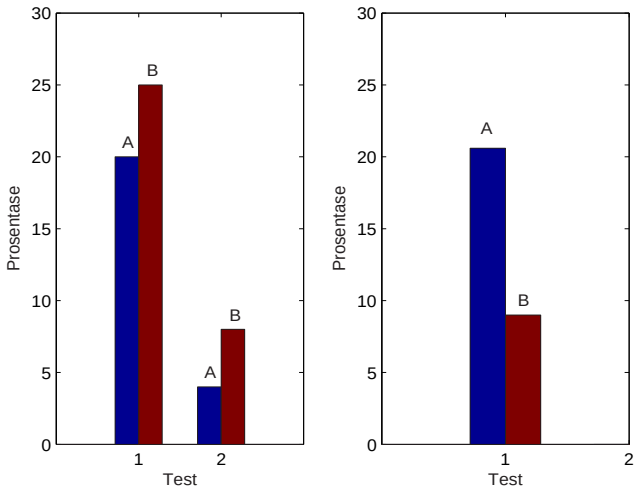

Figure 4.1: Ratio game dengan prosentase.

Sebenarnya kedua alternatif di atas kurang tepat. Pada alternatif 1, jumlah total pengujian yang dilakukan pada kedua sistem digunakan sebagai basis padahal jumlah pengujian tidak sama antara sistem A dan sistem B. Sedang alternatif kedua menggunakan jumlah kedua pengujian sebagai basis, yang mana keduanya juga berbeda.

Seringkali prosentase digunakan dengan cara yang salah. Misalnya, kita lebih senang mengatakan 83.3% dari perguruan tinggi di kota ini menggunakan Sistem X, daripada mengatakan 5 dari 6 perguruan tinggi menggunakan Sistem X.

Strategi Memenangkan Ratio Game *Note:*

Berikut ini adalah berapa guideline yang dapat digunakan untuk menggunakan ratio game.

1. *Jika sebuah sistem lebih baik pada semua benchmark, maka mengkontradiksikan konklusi dengan ratio game tidak dapat digunakan.* Kontradiksi solusi berarti bahwa satu sistem menunjukkan unjuk kerja yang bagus dengan menggunakan satu basis, sedangkan sistem yang lain me-

nunjukkan unjuk kerja yang bagus dengan menggunakan basis yang lain. Perhatikan pada Tabel 4.1 dimana sistem A bagus pada benchmark tertentu dan jelek pada benchmark yang lain. Pada contoh ini ratio game dapat digunakan untuk mengkontradiksikan konklusi. Perhatikan contoh pada Tabel 4.6 dibawah ini:

Table 4.6: Perbandingan Waktu Eksekusi Dua Buah Sistem

Raw Measurement			With A as a Base		With B as a Base	
	System		System		System	
Bencmark	A	B	A	B	A	B
I	0.50	1.00	1.00	2.00	0.50	1.00
J	1.00	1.50	1.00	1.50	0.67	1.00
Average	0.75	1.25	1.00	1.75	0.58	1.0

Pada tabel di atas terlihat bahwa Sistem A memberikan unjuk kerja lebih baik dari Sistem B untuk semua benchmark, sehingga dengan menggunakan ratio game terlihat bahwa sistem A juga lebih baik dari Sistem B dengan menggunakan basis yang manapun.

2. *Meskipun suatu sistem lebih baik dari yang lain untuk semua benchmark, unjuk kerja relatif yang lebih baik dapat diperlihatkan dengan memilih basis yang sesuai.* Sekalipun Tabel 4.6 menunjukkan bahwa Sistem A lebih baik dari Sistem B untuk berbagai basis, dengan prosentase dapat ditunjukkan bahwa Sistem A 40% lebih baik dari sistem B menggunakan raw data, 43% lebih baik dengan menggunakan A sebagai basis dan 42% lebih baik dengan menggunakan sistem B sebagai basis. Dalam hal ini menggunakan raw data lebih disukai.

3. *Jika sebuah sistem lebih baik untuk benchmar tertentu dan jelek untuk benchmark yang lain mengkontradiksikan konklusi dengan menggunakan*

ratio game hanya dapat digunakan untuk beberapa kasus saja (tidak semua). Karena itu uji coba untuk semua kasus harus dilakukan.

4. *Jika satuan ukuran unjuk kerja adalah LB (Lower is Better), maka lebih menggunakan sistem anda sebagai basis.* Sebagai contoh pada Tabel 4.6 satuan ukuran adalah waktu eksekusi, yaitu satuan LB, maka desainer sistem A lebih baik menggunakan sistem A sebagai basis.

5. *Jika satuan ukuran unjuk kerja adalah HB (Higher is Better), maka lebih menggunakan sistem lawan sebagai basis.* Misalnya satuan ukuran throughput, efisiensi, MPIS, MFLOPS adalah contoh-contoh satuan HB. Pada satuan unjuk kerja ini, sistem A akan memberikan rasio rata-rata lebih tinggi jika menggunakan sistem B sebagai basis.

6. *Benchmark yang memberikan unjuk kerja lebih baik sebaiknya diperpanjang, sedang benchmark yang memberikan unjuk kerja lebih jelek sebaiknya diperpendek.* Durasi waktu untuk masing-masing benchmark seringkali dapat diatur, jika sistem A memberi unjuk kerja lebih baik dari sistem B untuk kelanjutan pengujian sebaiknya durasi waktu pengujian sistem A diperpanjang, sebaiknya durasi waktu pengujian sistem B diperpendek.

Chapter 5

Membandingkan Sistem dengan Data Sampel

Kata sampel dalam bahasa Indonesia berasal dari kata bahasa Inggris *sample* atau *example*, yang keduanya berasal dari bahasa perancis kuno *essample*. Jadi pada dasarnya *sampel* adalah *example*. Untuk membuktikan sebuah teori atau menguji satu atau lebih sistem, tentu saja satu buah example tidak cukup, karena itu dibutuhkan beberapa sampel. Bab ini membahas tentang bagaimana menggunakan data sampel untuk membandingkan dua atau lebih sistem.

Sampel vs Populasi

Note:

- Misalkan kita menciptakan beberapa juta bilangan acak dengan menggunakan komputer dengan properti misalnya: mean = μ dan standar deviasi = σ. Jika kemudian semua bilangan tersebut kita letakkan dalam sebuah kotak dan sebuah sampel dengan observasi sebanyak n kita lakukan, maka akan terdapat sampel $\{x_1, x_2, \ldots, x_n\}$ dengan rata-rata sampel $\bar{x}$.
- Rata-rata sampel $\bar{x}$ tentunya berbeda dengan μ. Untuk membedakan, $\bar{x}$ disebut sebagai *rata-rata sampel* dan μ disebut sebagai *rata-rata populasi*.
- Dalam dunia nyata, karakteristik dari populasi sebenarnya tidak diketahui. Tugas dari seorang analis pada dasarnya adalah melakukan es-

timasi terhadap karakteristik dari populasi tersebut. Karena itu agar hasil estimasi mendekati populasi, maka percobaaan harus dilakukan sebanyak-banyaknya.

- Karakteristik dari populasi disebut sebagai *parameter*, sedangkan sampel estimasi disebut sebagai *statistik*.
- Untuk membedakan membedakan antara parameter dan statistik, maka simbol μ (rata-rata) dan σ (standar deviasi) digunakan untuk parameter. Sedangkan simbol $\bar{x}$ (rata-rata) dan s (standar deviasi) digunakan untuk statistik.

Interval Kepercayaan untuk Mean

Note:

Seperti dijelaskan di atas bahwa setiap rata-rata sampel merupakan estimasi terhadap rata-rata populasi. Jika kita mengambil sampel sebanyak k, maka kita juga akan memperoleh estimasi sebanyak k. Permasalahannya sekarang, bagaimana mencari satu nilai estimasi terhadap populasi dari k sample tersebut?

Pada kenyataannya tidak mungkin mendapatkan estimasi sempurna terhadap populasi. Karena itu hal yang bisa kita lakukan adalah dengan mengambil batasan probabilitas seperti berikut:

$$\text{Probabilitas}\{c_1 \leq \mu \leq c_2\} = 1 - \alpha \tag{5.1}$$

yang berarti: terdapat probabilitas tinggi, $1 - \alpha$, bahwa rata-rata populasi berada pada interval (c_1, c_2).

Interval (c_1, c_2) disebut sebagai interval kepercayaan (confidence interval) bagi rata-rata populasi, α disebut sebagai level signifikan (significance level), $100(1 - \alpha)$ disebut sebagai level/tingkat kepercayaan (confidence level) dan $1 - \alpha$ disebut sebagai koefisien kepercayaan (confidence coefficient). Umumnya tingkat kepercayaan secara tradisional diasumsikan bernilai 90 atau 95%; sehingga level signifikan α bernilai 0.1 atau 0.05.

Dengan adanya *central limit theory* kita tidak perlu melakukan terlalu banyak pengambilan sampel. Teori ini mengatakan bahwa: jika observasi terhadap sampel $\{x_1, x_2, \ldots, x_n\}$ bersifat bebas dan berasal dari populasi yang sama yang memiliki rata-rata μ dan standar deviasi σ, maka rata-rata sampel dengan jumlah sampel cukup besar akan mendekati kurva distribusi normal dengan rata-rata μ dan standar deviasi $\sigma/\sqrt{n}$:

$$\bar{x} \sim N(\mu, \sigma/\sqrt{n}) \tag{5.2}$$

Standar deviasi dari rata-rata sampel disebut sebagai *standard error* karena nilai dari standar deviasi tersebut berbeda dengan standar deviasi dari populasi. Dari rumus di atas terlihat jika jumlah n meningkat, maka standard error akan semakin kecil.

Dengan menggunakan teori central limit, interval kepercayaan untuk rata-rata populasi diberikan oleh:

$$(\bar{x} - z_{1-\alpha/2}s/\sqrt{n}, \bar{x} + z_{1-\alpha/2}s/\sqrt{n} \tag{5.3}$$

dimana $\bar{x}$ adalah rata-rata sampel, s adalah standar deviasi sampel, n adalah jumlah sample, dan $z_{1-\alpha/2}$ adalah quantile ke-$(1-\alpha/2)$ dari distribusi normal dengan standar deviasi 1. Nilai dari quantile ini dapat dilihat dalam Appendix B.

Contoh:

Jika diketahui sebuah sampel dari waktu CPU dalam percobaan yang dilakukan sebanyak 32 kali, sebagai berikut: { 3.1, 4.2, 2.8, 5.1, 2.8, 4.4, 5.6, 3.9, 3.9, 2.7, 4.1, 3.6, 3.1, 4.5, 3.8, 2.9, 3.4, 3.3, 2.8, 4.5, 4.9, 5.3, 1.9, 3.7, 3.2, 4.1, 5.1, 3.2, 3.9, 4.8, 5.9, 4.2}. Didapatkan bahwa rata-rata sampel $\bar{x} = 3.90$, standar deviasi $s = 0.95$ dan $n = 32$. Maka interval kepercayaan 90% dari rata-rata adalah:

$$\begin{aligned} &= 3.90 \mp (1.645)(0.95)/\sqrt{32} \\ &= (3.62, 4.17) \end{aligned} \tag{5.4}$$

Yang berarti bahwa: dengan kepercayaan 90% kita dapat mengatakan bahwa rata-rata populasi terletak pada nilai 3.62 dan 4.17. Kemungkinan salah

adalah 10%. Ini berarti jika kita mengambil sebanyak 100 sampel maka terdapat 90 sampel yang merupakan rata-rata populasi. Dengan cara yang sama, hitunglah rata-rata untuk interval kepercayaan 95% dan 99% !

Pada contoh di atas, interval kepercayaan hanya berlaku untuk jumlah sampel yang cukup besar yaitu lebih besar dari 30. Untuk jumlah sampel yang kecil, interval kepercayaan hanya dapat dibuat jika observasi berasal dari populasi yang memiliki distribusi normal. Interval kepercayaan untuk rata-rata populasi diberikan oleh:

$$(\bar{x} - t_{1-\alpha/2;n-1}s/\sqrt{n}, \bar{x} + t_{1-\alpha/2;n-1}s/\sqrt{n}) \tag{5.5}$$

dimana $t_{1-\alpha/2;n-1}$ adalah quantile ke-$(1-\alpha/2)$ dari distribusi t dengan derajad kebebasan $n-1$.

Contoh:

Misalkan kesalahan dari sebuah model didapatkan sebagai berikut: -0.04, -0.19, 0.14, -0.09, -0.14, 0.19, 0.04 dan 0.09. Nilai kesalahan ini memiliki distribusi normal dengan rata-rata=0 dan standar deviasi=0.138. Maka berdasarkan tabel dalam Appendix C didapatkan nilai dari $t_{[0.95;7]}$ adalah 1.895. Maka interval kesalahan dari rata-rata kesalahan adalah:

$$0 \mp 1.895 \times 0.138/\sqrt{8} = 0 \mp 0.0926 = (-0.0926, 0.0926)$$

Zero Mean Testing (Null Hypothesis)

Note:

Penggunaan yang paling umum untuk interval kepercayaan adalah untuk menguji apakah sebuah nilai terukur berbeda dari nol. Jika sebuah nilai hasil dari pengukuran sesuai dengan hasil pengujian pada kondisi lebih besar atau sama dengan level kepercayaan, $100(1-\alpha)\%$, maka nilai tersebut berbeda secara significan dari nol.

Pengujian dilakukan dengan cara menentukan interval kepercayaan dan menentukan jika nilai nol termasuk dalam interval tersebut. Terdapat 4 kemungkinan seperti ditunjukkan dalam Gambar 5.1 di bawah ini:

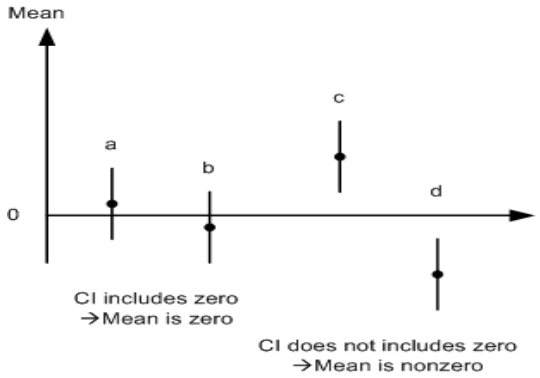

Figure 5.1: Zero mean test.

Pada kasus (a) dan (b), nilai nol masuk dalam interval kepercayaan (confidence interval - CI), berarti nilai terukur tidak berbeda dari nol secara signifikan. Sedang pada kasus (c) dan (d) nilai nol tidak masuk dalam interval kepercayaan, berarti nilai terukur berbeda secara signifikan dari nol.

Contoh:

Perbedaan waktu processor dari dua implementasi yang berbeda diukur pada tujuh workload. Didapatkan nilai perbedaannya adalah: {1.5, 2.6, -1.8, 1.3, -0.5, 1.7, 2.4}. Dapatkah kita mengatakan dengan 99% kepercayaan bahwa implementasi yang satu lebih unggul dari yang lain?

$$\begin{aligned} n &= 7 \\ \text{mean} &= 7.20/7 = 1.03 \\ \text{sample standard deviasi} &= \sqrt{2.57} = 1.60 \\ \text{interval kepercayaan} &= 1.03 \mp t \times 1.60/\sqrt{7} = 1.03 \mp 0.605t \\ 100(1-\alpha) &= 99; \\ \alpha &= 0.01; \\ 1-\alpha/2 &= 0.995; \end{aligned} \tag{5.6}$$

dengan menggunakan tabel dalam C, nilai t dengan 6 derajad kebebasan adalah $t_{[0.995;6]} = 3.707$, sehingga:

$$99\% \text{ interval kepercayaan} = (-1.21, 3.27) \tag{5.7}$$

Perhatikan bahwa nilai nol termasuk dalam itnerval kepercayaan. Berarti kita tidak dapat mengatakan dengan 99% kepercayaan bahwa nilai rata-rata dari perbedaan waktu processor di atas berbeda secara signifikan dari nol. Dengan kata lain kedua sistem tersebut tidak berbeda secara signifikan.

Implementasi dengan MATLAB:

```
>> x = [1.5 2.6 -1.8 1.3 -0.5 1.7 2.4]
x =
    1.5000    2.6000   -1.8000    1.3000   -0.5000    1.7000    2.4000

\%[h,p,ci,stats] = ttest(x,m,alpha) untuk n< 30 \\
\%[h,sig,ci,zval]= ztest(x,m,sigma,alpha) untuk n>= 30

>> [h,p,ci,stats]=ttest(x,1.03,0.01)
h =
     0
p =
    0.9982
ci =
   -1.2189    3.2760
stats =
    tstat: -0.0024
       df: 6
       sd: 1.6039
```

Membandingkan Dua Observasi

Note:

Untuk membandingkan dua buah hasil pengukuran dapat dilakukan dengan dua cara, yaitu: observasi berpasangan (paired observation) dan observasi tidak berpasangan (unpaired observation).

Observasi Berpasangan

Apabila kita melakukan sebanyak n eksperimen pada dua buah sistem sehingga terhadap hubungan one-to-one pada setiap pengujian ke-i pada sistem A dan pengujian ke-i pada sistem B, maka observasi yang kita lakukan disebut sebagai berpasangan. Misalnya, x_i dan y_i merupakan performance dari workload ke-i, maka observasi akan menjadi berpasangan.

Analisa terhadap dua buah observasi yang berpasangan maka pasangan ke-n didapatkan dengan cara mencari selisih dari kedua observasi. Selanjutnya, dengan menggunakan interval kepercayaan kita uji dengan menggunakan zero mean test.

Contoh:

Enam buah workload digunakan pada dua buah sistem. Dari hasil observasi didapatkan data sebagai berikut: {(5.4,19.1), (16.6,3.5), (0.6,3.4), (1.4,2.5), (0.6,3.6), (7.3,1.7)}. Apakah sistem yang satu lebih baik dari sistem yang lain?

Perbedaan performance adalah: {-13.7, 13.1, -2.8, -1.1, -3.0, 5.6}.

$$
\begin{aligned}
n &= 6 \\
\text{mean} &= -0.32 \\
\text{sample standard deviasi} &= 9.03 \\
\text{interval kepercayaan} &= -0.32 \mp t \times 9.03/\sqrt{6} \\
&= -0.32 \mp 3.69t \\
100(1-\alpha) &= 90; \\
\alpha &= 0.1; \\
1-\alpha/2 &= 0.95; \qquad (5.8) \\
90\% \text{ interval kepercayaan} &= (-7.75, 7.11) \qquad (5.9)
\end{aligned}
$$

Karena nilai nol termasuk dalam interval kepercayaan, maka kedua sistem tersebut tidak berbeda.

Implementasi dengan MATLAB:

```
>> x = [5.4 16.6 0.6 1.4 0.6 7.3]

x =
    5.4000   16.6000    0.6000    1.4000    0.6000    7.3000

>> y = [19.1 3.5 3.4 2.5 3.6 1.7]

y =
   19.1000    3.5000    3.4000    2.5000    3.6000    1.7000

>> [h,p,ci,stats]=ttest(x,y,0.1)
h =
     0
p =
    0.9349
ci =
   -7.7488    7.1155
stats =
    tstat: -0.0859
       df: 5
       sd: 9.0345
```

Observasi Tidak Berpasangan

Misalkan kita memiliki dua sampel dengan ukuran n_a dan n_b yang diambil dari sistem A dan B. Observasi disebut tidak berpasangan dalam artian tidak ada korespondensi antara observasi ke-i dari kedua sampel. Untuk menentukan interval kepercayaan dari perbedaan performance rata-rata dari kedua sistem

dibutuhkan *estimasi terhadap variance dan jumlah derajad kebebasan efektif*. Prosedur pengujian observasi tidak berpasangan adalah sebagai berikut (seluruh prosedur ini disebut sebagai t-test):

1. Hitung rata-rata masing-masing sampel:

$$\bar{x}_a = \frac{1}{n_a}\sum_{i=1}^{n_a} x_{ia} \tag{5.10}$$

$$\bar{x}_b = \frac{1}{n_b}\sum_{i=1}^{n_b} x_{ib} \tag{5.11}$$

2. Hitung standar deviasi dari sampel:

$$s_a = \left\{ \frac{(\sum_{i=1}^{n_a} x_{ia}^2) - n_a \bar{x}_a^2}{n_a - 1} \right\}^{1/2} \tag{5.12}$$

$$s_b = \left\{ \frac{(\sum_{i=1}^{n_b} x_{ib}^2) - n_b \bar{x}_b^2}{n_b - 1} \right\}^{1/2} \tag{5.13}$$

3. Hitung selisih dari rata-rata sampel:

$$\bar{x}_a - \bar{x}_b \tag{5.14}$$

4. Hitung standar deviasi dari selisih rata-rata sampel:

$$s = \sqrt{\frac{s_a^2}{n_a} + \frac{s_b^2}{n_b}} \tag{5.15}$$

5. Hitung jumlah derajad kebebasan efektif:

$$\nu = \frac{(s_a^2/n_a + s_b^2/n_b)^2}{\frac{1}{n_a+1}\left(\frac{s_a^2}{n_a}\right)^2 + \frac{1}{n_b+1}\left(\frac{s_b^2}{n_b}\right)^2} - 2 \tag{5.16}$$

6. Hitung interval kepercayaan dari selisih rata-rata:

$$(\bar{x}_a - \bar{x}_b) \mp t_{[1-\alpha/2;\nu]} s \tag{5.17}$$

dimana $t_{[1-\alpha/2;\nu]}$ adalah quantile ke-$(1-\alpha/2)$ dari t-variate dengan derajad kebebasan ν.

7. Jika nol termasuk dalam interval kepercayaan, maka perbedaan antara kedua sistem tidak signifikan pada $100(1-\alpha)\%$ level kepercayaan. Jika nol tidak termasuk dalam interval kepercayaan, maka tanda dari selisih rata-rata tersebut menunjuk pada sistem mana yang lebih baik.

Contoh:

Waktu prosesor yang dibutuhkan untuk melakukan eksekusi terhadap sebuah tugas diukur pada dua sistem. Waktu dari sistem A adalah { 5.36, 16.57, 0.62, 1.41, 0.64, 7.26}. Waktu dari sistem B adalah { 19.12, 3.52, 3.38, 2.50, 3.60, 1.74}. Apakah kedua sistem tersebut berbeda secara signifikan pada 90% level kepercayaan?

Untuk sistem A:

$$\begin{aligned} \text{mean}\bar{x}_a &= 5.31 \\ \text{varian}s_a^2 &= 37.92 \\ n_a &= 6 \end{aligned}$$

Untuk sistem B:

$$\begin{aligned} \text{mean}\bar{x}_b &= 5.64 \\ \text{varian}s_b^2 &= 44.11 \\ n_b &= 6 \end{aligned}$$

Maka dapat dihitung:

$$\begin{aligned} \bar{x}_a - \bar{x}_b &= -0.33 \\ s &= 3.698 \\ \nu &= 11.921 \\ \alpha &= 0.1 \\ t_{[1-\alpha/2;\nu]} &= 1.71 \\ \text{interval kepercayaan pada } 90\% &= (-6.92, 6.26) \end{aligned}$$

Karena nilai nol masuk dalam interval kepercayaan maka pada level kepercayaan 90% dapat dikatakan bahwa kedua sistem tidak berbeda.

Implementasi dengan MATLAB:

```
>> x = [5.36 16.57 0.62 1.41 0.64 7.26]
x =
    5.3600   16.5700    0.6200    1.4100    0.6400    7.2600

>> y = [19.12 3.52 3.38 2.50 3.60 1.74]
y =
   19.1200    3.5200    3.3800    2.5000    3.6000    1.7400

>> [h,sig,ci,stat]=ttest2(x,y,0.1)

h =
     0
sig =
    0.9299
ci =
   -7.0350    6.3683
stat =
    tstat: -0.0902
       df: 10
       sd: 6.4043
```

Hasil perhitungan dari MATLAB berbeda dengan perhitugnan manual karena MATLAB mengasumsikan bahwa kedua sampel memiliki nilai standar deviasi yang sama, tidak demikian pada perhitungan manual.

Chapter 6

Membangun Model dengan Linear Regression

Salah satu teknik yang dapat digunakan untuk membangun sebuah model jika data sampel diketahui adalah dengan menggunakan *Linear Regression*. Linear regression dapat dipandang sebagai estimasi terhadap pola dari data sample yang cenderung linear. Jika dilakukan plotting terhadap data sampel, maka linear regression akan berupa garis tren yang mendekati tren/pola dari data sampel.

Secara matematis, model linear ini dinyatakan sebagai:

$$\hat{y} = b_0 + b_1 x \tag{6.1}$$

dimana $\hat{y}$ adalah *predicted response* dan x adalah *predictor variable*. Sedangkan parameter b_0 dan b_1 disebut sebagai *parameter regresi* yang ditentukan dari sampel data. Jika diberikan observasi sebanyak n dengan pasangan $(x_1, y_1), \ldots, (x_n, y_n)$ maka parameter regresi didefinisikan oleh persamaan:

$$b_1 = \frac{\sum xy - n\bar{x}\bar{y}}{\sum x^2 - n(\bar{x})^2} \tag{6.2}$$

dan

$$b_0 = \bar{y} - b_1\bar{x} \tag{6.3}$$

dimana $\bar{x}$ adalah nilai rata-rata dari predictor variable dan $\bar{y}$ adalah nilai rata-rata dari response.

Contoh:

Jumlah perangkat I/O dari disk yang terpasang pada komputer dan waktu

proses processor pada 7 macam program diukur sebagai berikut:{(14,2),(16,5), (27,7),(42,9),(39,10,(50,13),(83,20) }. Tentukan model linear dari waktu CPU sebagai fungsi dari perangkat I/O dari disk!

Dari data tersebut diketahui $n = 7$, $\sum xy = 3375$, $\sum x = 271$, $\sum x^2 = 13,855$, $\sum y = 66$, $\sum y^2 = 828$, $\bar{x} = 38.71$,$\bar{y} = 9.43$, sehingga:

$$\begin{aligned} b_1 &= \frac{3375 - 7 \times 38.71 \times 9.43}{13,855 - 7 \times (38.71)^2} = 0.2438 \\ b_0 &= 9.43 - 0.2438 \times 38.71 = -0083 \end{aligned}$$

Sehigga linear model yang diinginkan adalah:
Waktu CPU = -0.0083 + 0.2438 (jumlah dari I/O).

Plotting dari linear model ditunjukkan dalam Gambar 6.1 di bawah ini:

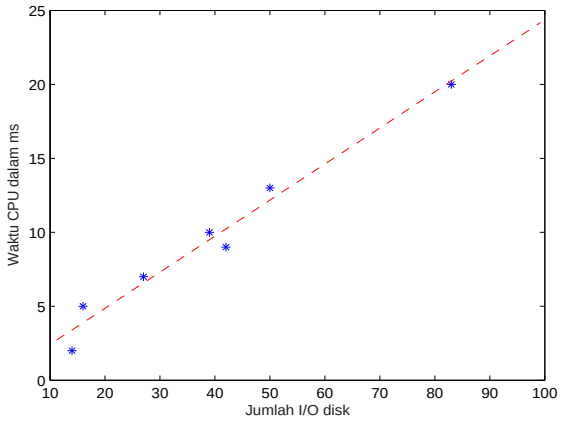

Figure 6.1: Linear model waktu CPU sebagai fungsi dari dari I/O.

Implementasi dengan MATLAB:

```
% Linear Regression

x = [14 16 27 42 39 50 83]; % Predictor
y = [2 5 7 9 10 13 20]; % Response

[m,n] = size(x);
```

```
% sigma xy
sigma_xy = sum(x.*y);

% sigma x
sigma_x = sum(x);

% sigma x^2
sigma_x2 = sum(x.^2);

% sigma y
sigma_y = sum(y);

% sigma y^2
sigma_y2 = sum(y.^2);

% mean x
mean_x = mean(x);

% mean y
mean_y = mean(y);

% Regression parameters
b1 = (sigma_xy - n*mean_x*mean_y)/(sigma_x2 - n*mean_x^2)
b0=mean_y- b1*mean_x

% estimation
y_hat = b0 + b1*x
sigma_y_hat = sum(y_hat)
error = y - y_hat
sigma_error = sum(error)
```

```
error2 = error.^2

sigma_error2 =sum(error2)

%Plot
x_minimum = min(x) - 0.2*min(x);
x_maximum = max(x) + 0.2*max(x);
x_hat_plot = x_minimum:x_maximum;
y_hat_plot = b0 + b1*x_hat;

plot(x,y,'*'), hold on
plot(x_hat_plot,y_hat_plot,'r--')
xlabel('Jumlah I/O disk');ylabel('Waktu CPU dalam ms')
```

Interval Kepercayaan untuk Parameter Regresi

Note:

Seperti terlihat dalam bagian terdahulu bahwa parameter b_0 dan b_1 merupakan estimasi dari sampel dengan jumlah n. Jika kita menggunakan sampel yang berbeda mungkin akan menghasilkan nilai yang berbeda pula. Karena kita dapat menggunakan interval kepercayaan untuk memberikan estimasi terhadap populasi yang sesungguhnya (model yang sebenarnya). Model yang sebenarnya didefinisikan sebagai:

$$y = \beta_0 + \beta_1 x. \tag{6.4}$$

Parameter regresi b_0 dan b_1 adalah statistik daro parameter populasi β_0 dan β_1.

Standar deviasi dari b_0 dan b_1 dapat dihitung dengan rumusan:

$$s_{b_0} = s_e \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum x^2 - n\bar{x}^2}\right]^{1/2} \tag{6.5}$$

$$s_{b1} = \frac{s_e}{[\sum x^2 - n\bar{x}^2]^{1/2}} \tag{6.6}$$

dimana s_e adalah standar deviasi dari error. Maka inverval kepercayaan $100(1-$

$\alpha)\%$ untuk b_0 dan b_1 dapat dihitung dengan $t_{[1-\alpha/2;n-2]}$ adalah:

$$\text{interval kepercayaan} = b_0 \mp t_{[1-\alpha/2;n-2]} s_{b_0} \tag{6.7}$$

$$\text{interval kepercayaan} = b_1 \mp t_{[1-\alpha/2;n-2]} s_{b_1} \tag{6.8}$$

Untuk contoh kasus di atas di dapatkan bahwa $n = 7$, $\sum x^2 = 13,855$, $\bar{x} = 38.71$,$s_e = 1.0834$. Dari nilai tersebut kita dapatkan standar deviasi dari b_0 dan b_1 adalah:

$$s_{b_0} = 1.0834 \left[\frac{1}{7} + \frac{(38.71)^2}{13,855 - 7 \times 38.71^2}\right]^{1/2} = 0.8311 \tag{6.9}$$

$$s_{b_1} = \frac{1.0834}{[13,855 - 7 \times 38.71^2]^{1/2}} = 0.0187 \tag{6.10}$$

Maka dari tabel pada C didapatkan t-variate dengan 5 derajad kebebasan dari interval kepercayaan 90% adalah 2.015. Sehingga interval kepercayaan 90% dari b_0 adalah:

$$-0.0083 \mp (2.015)(0.8311) = -0.0083 \mp 1.6747 = (-1.6830, 1.6663)$$

Dan interval kepercayaan 90% dari b_1 adalah:

$$-0.2438 \mp (2.015)(0.0187) = -0.2438 \mp 0.0376 = (0.2061, 0.2814).$$

Implementasi dengan MATLAB:

```
>> x = [14 16 27 42 39 50 83]

x =
    14    16    27    42    39    50    83

>> y = [2 5 7 9 10 13 20]

y =
     2     5     7     9    10    13    20
```

```
% polytool(x,y,N,alpha,xname,yname), N adalah derajad dari polynomial,
% karena kita mengimplementasikan regresi linear maka N=1;

>> polytool(x,y,1,0.1,'Jumlah IO Disk','Waktu CPU dalam ms')
>> betaci

betaci =

    0.2061   -1.6830
    0.2814    1.6664
```

Hasil plotting implementasi dari MATLAB ditunjukkan dalam Gambar di bawah ini:

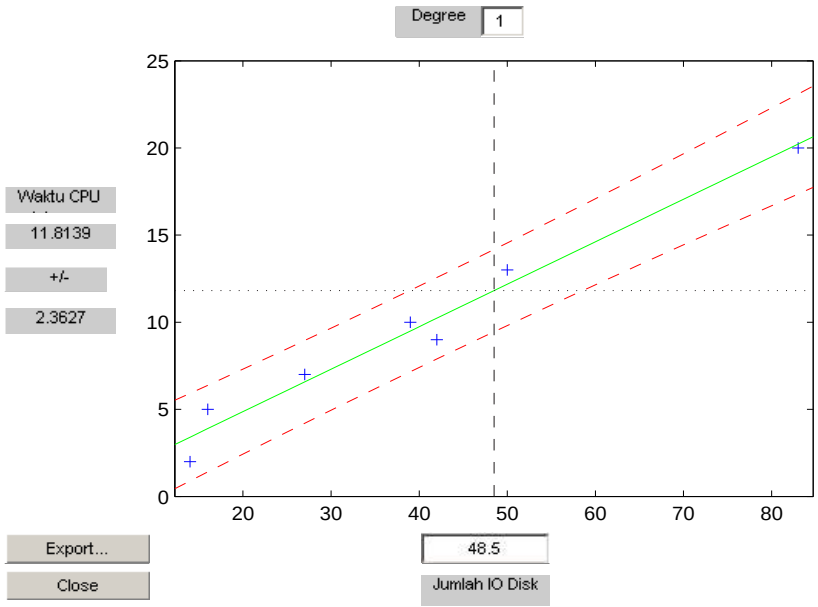

Figure 6.2: Linear model waktu CPU sebagai fungsi dari dari I/O dengan interval kepercayaan 90%.

Chapter 7

Teori Antrian

Prinsip kerja dari sebuah sistem komputer adalah: beberapa pekerjaan diproses dengan menggunakan resource yang sama. Misalnya, beberapa pekerjaan pada satu waktu tertentu membutuhkan layanan dari CPU, disk atau perangkat-perangkat (devais) I/O yang lain. Akan tetapi secara prinsip hanya ada satu pekerjaan yang dapat dilayani oleh sebuah devais, sementara pekerjaan yang lain dalam kondisi antrian. Teori antrian akan sangat berguna dalam menentukan waktu yang dibutuhkan oleh setiap pekerjaan dalam sebuah antrian. Selain itu, teori ini juga dapat digunakan untuk memprediksi, misalnya waktu respons dari sebuah sistem, dsb. Karena itu model-model antrian seringkali juga digunakan dalam analisa unjuk kerja dari sistem komputer.

Komponen dasar dari sebuah antrian dapat digambarkan dengan model yang sederhana seperti ditunjukkan dalam Gambar 7.1. Model ini menggambarkan antrian mahasiswa untuk menggunakan terminal internet yang jumlahnya terbatas, sedang jumlah mahasiswa tak terbatas. Jika semua terminal dalam kondisi terpakai, maka mahasiswa yang datang akan dimasukkan dalam sebuah antrian. Dalam teori antrian, mahasiswa disebut sebagai *customer*. Beberapa istilah penting dalam analisa antrian sistem dijelaskan sebagai berikut:

1. *Proses kedatangan (Arrival Process):* Jika mahasiswa tiba pada waktu $t_1, t_2, \ldots, t_j$, maka bilangan random $\tau_j = t_j - t_{j-1}$ disebut sebagai *waktu antar kedatangan.* Biasanya diasumsikan bahwa τ_j adalah variabel random yang bersifat *independent and identically distributed (IID)*. Model paling umum untuk proses kedatangan adalah kedatangan Poisson. Hal

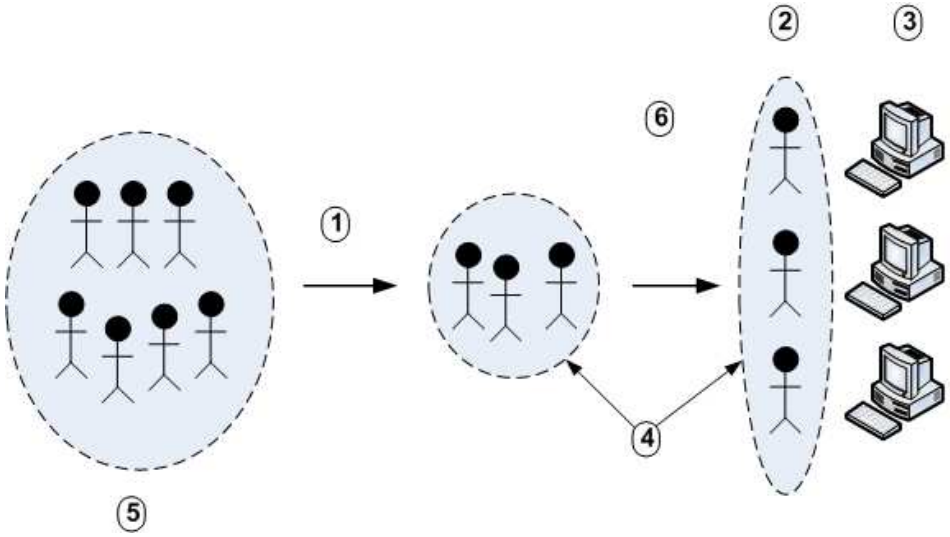

Figure 7.1: Queue model untuk antrian mahasiswa menggunakan komputer.

ini berarti bahwa waktu antar kedatangan diasumsikan memiliki distribusi eksponensial.

2. *Distribusi Waktu Layanan (Service Time Distribution):* Waktu layanan adalah waktu yang dibutuhkan oleh para mahasiswa untuk menggunakan terminal. Waktu layanan biasanya diasumsikan sebagai variabel random yang memiliki distribusi eksponensial.

3. *Jumlah Pemberi Layanan (Number of Servers):* adalah jumlah terminal yang dapat memberikan layanan kepada mahasiswa. Dalam teori antrian, pemberi layanan ini diasumsikan bahwa semuanya identik.

4. *Kapasitas Sistem (System Capacity):* Jumlah maksimum dari mahasiswa yang dapat menggunakan layanan dibatasi karena keterbatasan tempat dan juga untuk menghindari waktu tunggu yang panjang. Jumlah mahasiswa ini disebut sebagai kapasitas sistem.

5. *Ukuran Populasi: (Population Size)* adalah jumlah total mahasiswa yang diperbolehkan datang ke pusat komputer tersebut.

6. *Disiplin Layanan (Service Discipline):* Yang dimaksud dengan disiplin layanan adalah bagaimana urutan mahasiswa dilayani. Misalnya, FCFS

(First Come First Served), LCFS (Last Come First Served), dsb.

Untuk menspesifikasikan sebuah sistem antrian, keenam parameter di atas digunakan dengan menggunakan symbol: A/S/m/B/K/SD yang disebut sebagai *notasi Kendall*, di mana setiap parameter dengan keenam parameter seperti dijelaskan di atas. A adalah waktu antar kedatangan, S adalah distribusi waktu layanan, m adalah jumlah pemberi layanan, B adalah kapasitas sistem, K adalah ukuran populasi dan SD adalah disiplin layanan.

Distribusi dari waktu antar kedatangan biasanya disimbulkan dengan menggunakan salah satu simbol di bawah ini:

M	Eksponensial
E_k	Erlang dengan parameter k
H_k	Hyper-eksponensial dengan paramter k
D	Deterministik
G	General

Sebagai contoh, jika sebuah sistem antrian direpresentasikan dengan M/M/5/30/1200/FCFS berarti: waktu antar kedatangan memiliki distribusi ekponensial demikian pula distribusi waktu layanan. Sistem tersebut memiliki 5 buah pemberi layanan dan 30 tempat untuk antrian, yang terdiri atas 5 tempat untuk mereka yang sedang dilayani dan 25 tempat untuk menunggu. Secara keseluruhan ada sebanyak 1200 pekerjaan yang harus dilayani dengan menggunakan disiplin layanan first come first served.

Aturan-Aturan untuk Semua Antrian

Note:

Gambar 7.2 menunjukkan variabel-variabel kunci yang digunakan untuk analisa pada sistem antrian dan dijelaskan sebagai berikut:

τ = waktu antar kedatangan, waktu antara dua kedatangan yang berurutan.

λ = laju kedatangan rata-rata = $1/\mathrm{E}[\tau]$, dimana $\mathrm{E}[\tau]$ adalah rata-rata waktu antar kedatangan.

s = waktu layanan per pekerjaan.

μ = laju layanan rata-rata per server (pemberi layanan) = $1/\mathrm{E}[s]$, dimana $\mathrm{E}[s]$ adalah rata-rata waktu layanan per pekerjaan. Maka total layanan untuk m buah server adalah $m\mu$.

n = jumlah pekerjaan dalam sistem. Seringkali disebut juga sebagai *panjang antrian (queue length)*. Perhatikan bahwa n termasuk pekerjaan yang sedang menerima layanan saat itu dan pekerjaan yang sedang menunggu dalam antrian.

n_q = jumlah pekerjaan yang sedang menunggu dalam antrian (menunggu untuk menerima layanan).

n_s = jumlah pekerjaan yang sedang menerima layanan.

r = waktu respons atau waktu selama pekerjaan berada di dalam sistem. Termasuk waktu yang dibutuhkan oleh pekerjaan untuk menunggu layanan dan waktu yang dibutuhkan oleh perkerjaan untuk menerima layanan.

w = waktu tunggu, yaitu interval dari waktu kedatangan sampai saat dimana pemberian layanan dimulai.

Kondisi Stabilitas

Jika jumlah pekerjaan yang harus dikerjakan bertambah terus-menerus sampai tak terbatas, maka sistem akan menjadi tidak stabil. Untuk dapat menacapai stabilitas sistem maka dibutuhkan agar laju kedatangan rata-rata (λ) kurang

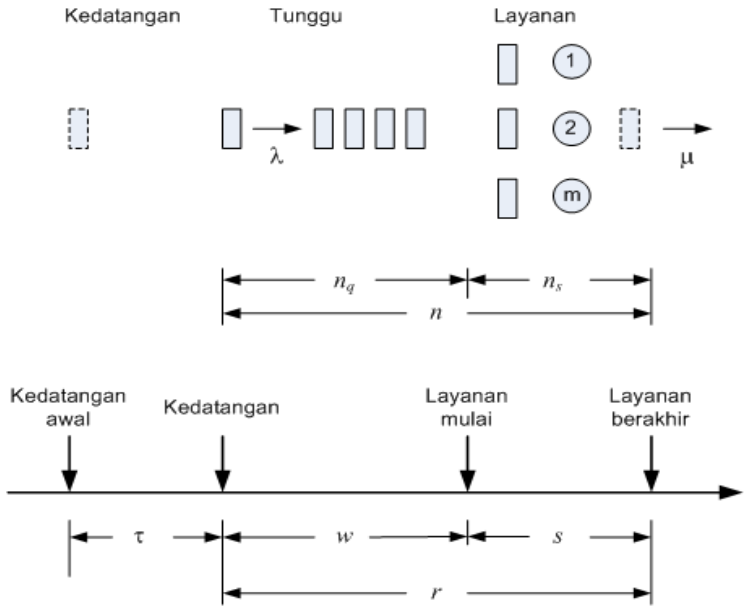

Figure 7.2: Variabel kunci dalam teori antrian.

dari laju layanan rata-rata, sehingga:

$$\lambda < m\mu. \tag{7.1}$$

Little's Law

Jika tidak pekerjaan yang hilang akibat ketidakcukupan memori, maka jumlah pekerjaan rata-rata di dalam sistem berkaitan dengan waktu respons rata-rata dari sistem tersebut.

$$\text{Jumlah pekerjaan rata-rata dalam sistem} = \text{laju kedatangan} \times \text{waktu respons rata-rata}. \tag{7.2}$$

dengan cara yang sama:

$$\text{Jumlah pekerjaan rata-rata dalam antrian} = \text{laju kedatangan} \times \text{waktu tunggu rata-rata}. \tag{7.3}$$

Kedua persamaan di atas disebut sebagai Little's Law.

Misalkan kita sedang memonitor sebuah sistem dalam interval waktu T dan mencatat setiap kedatangan dan keluaran dari setiap pekerjaan. Jika T

diasumsikan besar sekali, maka jumlah kedatangan kira-kira akan sama dengan jumlah keberangkatan, yaitu N. Sehingga waktu kedatangan dapat dihitung dengan rumusan:

$$\text{laju kedatangan} = \text{total kedatangan/waktu total} = N/T \tag{7.4}$$

Sementara itu, waktu rata-rata selama pekerjaan berada di dalam sistem dapat dihitung dengan persamaan:

$$\text{waktu rata-rata di dalam sistem} = r/N \tag{7.5}$$

Jadi dengan kedua persamaan di atas little's law dapat diturunkan sebagai:

$$\begin{aligned} \text{Jumlah pekerjaan rata-rata dalam sistem} &= r/T \\ &= \frac{N}{T} \times \frac{r}{N} \\ &= \text{laju kedatangan} \times \text{waktu rata-rata di dalam sistem.} \end{aligned} \tag{7.6}$$

Dengan cara yang sama kita juga dapat membuat persamaan:

$$\begin{aligned} &\text{Jumlah pekerjaan rata-rata yang dilayani} = \\ &\text{laju kedatangan} \times \text{waktu layanan rata-rata.} \end{aligned} \tag{7.7}$$

Contoh

Pengamatan terhadap sebuah disk yang berfungsi sebagai pemberi layanan didapatkan bahwa waktu rata-rata untuk memenuhi permintaan dari I/O adalah 100 ms. Laju dari I/O adalah 100 permintaan per detik. Berapa jumlah permintaan rata-rata pada disk tersebut?

Dengan menggunakan Little's law, didapatkan bahwa:

$$\begin{aligned} &\text{Jumlah permintaan pada disk} = \\ &\text{laju kedatangan} \times \text{waktu respons} \\ &= (100 \text{ permintaan/detik}) \times (0.1 \text{ detik}) \\ &= 10 \text{ permintaan} \end{aligned}$$

Utilisasi Server

Utilisasi server (ρ) adalah jumlah pekerjaan rata-rata yang dapat ditangani oleh pemberi layanan yang didefinisikan oleh rumusan:

$$\text{utilisasi server}(\rho) = \text{laju kedatangan} \times \text{waktu layanan}. \tag{7.8}$$

Nilai dari utilisasi server selalu terletak antara 0 dan 1, jika tidak maka terjadi ketidakstabilan pada sistem dimana jumlah pekerjaan lebih banyak daripada yang dapat dilayani. Utilisasi disebut juga sebagai *traffic intensity*.

Contoh

Sebuah sistem I/O dengan satu buah disk memiliki rata-rata 50 permintaan per detik. Asumsikan waktu rata-rata bagi disk untuk memberikan layanan permintaan I/O adakah 10 ms. Berapa utilisasi dari sistem tersebut?

$$\text{utilisasi server}(\rho) = \text{laju kedatangan} \times \text{waktu layanan} = 50 \times 0.01 = 0.5.$$

Waktu Tunggu/Waktu Antri

Waktu tunggu adalah sejumlah waktu yang dibutuhkan oleh pekerjaan sebelum mendapatkan layanan, didefinisikan oleh persamaan:

$$w = s \times \frac{\rho}{1-\rho}. \tag{7.9}$$

Panjang Antrian

Panjang antrian adalah panjang rata-rata dari pekerjaan yang sedang di dalam antrian, didefinisikan oleh persamaan:

$$n_q = \frac{\rho^2}{1-\rho}. \tag{7.10}$$

Latihan

1. Sebuah prosesor dapat mengirimkan pekerjaan kepada 40 disk I/O per detik dimana permintaan ini terdistribusi secara eksponensial. Waktu layanan rata-rata dari disk tertua adalah 20 ms. Jawablah pertanyaan-pertanyaan di bawah ini:
   (a) Secara rata-rata berapa utilisasi dari disk ?
   (b) Berapa waktu rata-rata yang dihabiskan dalam antrian ?
   (c) Berapa respons waktu dari disk (termasuk waktu tunggu dan waktu layanan)?
   (d) Berapa % dari waktu dihabiskan untuk menunggu ?
2. Jika disk tersebut diganti dengan yang lebih cepat yang memiliki waktu layanan rata-rata 10 ms, hitunglah kembali utilisasi dari disk dan waktu tunggu! Berapa cepat waktu respons dari disk yang baru ini dibanding dengan disk lama?

M/M/1 Queue

Note:

Model antrian M/M/1 merupakan model yang sering digunakan. Jenis antrian ini dapat digunakan untuk memodelkan sistem dengan satu prosesor atau perangkat-perangkat tuggal yang ada di dalam komputer. Dengan menggunakan model ini, diasumsikan bahwa waktu antar kedatangan (interarrival times) dan waktu layanan (service time) terdistribusi secara ekponensial dan hanya ada satu buah server. Tidak ada buffer dan tidak ada batasan populasi. Disiplin layanan yang digunakan adalah FCFS. Untuk melakukan analisa terhadap tipe antrian ini, terdapat dua hal penting yang harus diketahui, yaitu: laju kedatangan rata-rata (λ) dan laju layanan rata-rata (μ). Secara ringkas parameter-parameter yang digunakan pada M/M/1 queue adalah sebagai berikut:

1. Parameter:

   λ = laju kedatangan pekerjaan per unit waktu,

   μ = laju layanan pekerjaan per unit waktu.

2. Traffic Intensity: $\rho = \lambda/\mu$.

3. Kondisi stabil: Traffic intensity ρ harus lebih kecil dari 1.

4. Probabilitas nol pekerjaan di dalam sistem: $p_0 = 1 - \rho$.

5. Probabilitas terdapat n pekerjaan di dalam sistem: $p_n = (1 - \rho)\rho^n$, $n = 0, 1, \ldots, \infty$.

6. Jumlah rata-rata pekerjaan di dalam sistem: $\mathrm{E}[n] = \rho/(1 - \rho)$.

7. Varian dari jumlah pekerjaan di dalam sistem: $\mathrm{Var}[n] = \rho/(1 - \rho)^2$.

8. Probabilitas terdapat k pekerjaan di dalam queue:

$$P(n_q = k) = \begin{cases} 1 - \rho^2 & k = 0 \\ (1 - \rho)\rho^{k+1} & k > 0. \end{cases}$$

9. Jumlah rata-rata pekerjaan di dalam queue: $\mathrm{E}[n_q] = \rho^2/(1 - \rho)$.

10. Varian dari jumlah pekerjaan di dalam queue: $\mathrm{Var}[n] = \rho^2(1 + \rho - \rho^2)/(1/\rho^2)$.

11. Fungsi distribusi kumulatif dari response waktu: $\mathrm{F}(r) = 1 - e^{-r\mu(1-\rho)}$.

12. Rata-rata response waktu: $\mathrm{E}[r] = (1/\mu)/(1 - \rho)$.

13. Varian dari response waktu: $\mathrm{Var}[r] = \frac{1/\mu^2}{(1-\rho)^2}$.

14. q-Percentile dari response waktu: $\mathrm{E}[r] \ln[100/(100 - q)]$.

15. 90%-Percentile dari response waktu: $2.3\mathrm{E}[r]$.

16. Fungsi distribusi kumulatif dari waktu tunggu: $\mathrm{F}(w) = 1 - \rho e^{-\mu w(1-\rho)}$.

17. Rata-rata waktu tunggu: $\mathrm{E}[w] = \rho\frac{1/\mu}{(1-\rho)}$.

18. Varian dari waktu tunggu: $\text{Var}[w] = (2-\rho)\rho/[\mu^2(1-\rho)^2]$.
19. q-Percentile dari waktu tunggu: $\max\left(0, \frac{\text{E}[w]}{\rho}\ln[100\rho/(100-q)]\right)$.
20. 90%-Percentile dari waktu tunggu: $\max\left(0, \frac{\text{E}[w]}{\rho}\ln[10\rho]\right)$.
21. Probabilitas untuk menemukan n atau lebih pekerjaan di dalam sistem: ρ^n.
22. Probabilitas untuk melayani n pekerjaan dalam waktu sibuk:

$$\frac{1}{n}\begin{pmatrix} 2n-2 \\ n-1 \end{pmatrix}\frac{\rho^{n-1}}{(1+\rho)^{2n-1}}.$$

Contoh

Pada sebuah gateway dari suatu jaringan terukur bahwa paket yang datang memiliki rata-rata 125 paket per detik (packet per second - pps) dan gateway tersebut menggunakan waktu 2 ms untuk mem-forward paket-paket itu. Dengan menggunakan M/M/1 lakukan anlisa terhadap gateway tersebut. Berapakah probabilitas terjadi buffer overflow apabila gateway memiliki 13 buffer? Berapa jumlah buffer yang dibutuhkan agar jumlah paket yang hilang dapat dijaga dibawah 1 paket per 1 juta?

Laju kedatangan (λ)= 125 pps.
Laju layanan (μ) = $1/0.002$ = 500 pps.
Utilisasi Gateway (ρ) = $\lambda/\mu = 0.25$.
Probabilitas terdapat n packet di dalam gateway = $(1-\rho)\rho^n = 0.75(0.25)^n$.
Jumlah rata-rata paket di dalam gateway = $\rho/(1-\rho) = 0.25/0.75 = 0.33$.
Rata-rata waktu tunggu di dalam gateway = $\rho\frac{1/\mu}{(1-\rho)} = \frac{1/500}{1-0.25} = 2.66ms$.
Probabilitas dari buffer overflow= P(lebih dari 13 paket dalam gateway)
$= \rho^{13} = 0.25^{13} = 1.49 \times 10^{-8}$
$\approx$ 15 paket per miliar paket.
Untuk membatasi agar probabilitas paket hilang kurang dari 10^{-6}, maka:

$\rho^n \leq 10^{-6}$

atau $n > \log(10^{-6})/\log(0.25) = 9.96$

dengan demikian dibutuhkan kira-kira 10 buffer.

Latihan

Bandingkan sebuah packet switch network dengan bandwidth 1000 Mbps dan 10 buah packet switch yang masing-masing memiliki bandwidth 100 Mbps. Asumsikan bahwa ukuran rata-rata dari paket adalah 250 bytes, laju kedatangan 250 ribu paket per detik, dan waktu antar kedatangan terdistribusi secara eksponensial. Berapakah rata-rata waktu response untuk masing-masing alternatif?

Chapter 8

Unjuk Kerja Layanan WEB

Bagian ini mendiskusikan hal-hal yang mempengaruhi unjuk kerja dari layanan-layanan WEB.

- Layanan-layanan WEB dipengaruhi oleh delay yang muncul sebagai akibat dari: delay pemrosesan layanan oleh web server dan delay pada jaringan, yaitu semua delay yang terjadi antara sisi client dan sisi web server.

- Server merupakan komponen utama pada internet dan intranet. Layanan-layanan web yang tersedia diberikan oleh berbagai macam server yang menggunakan berbagai macam komputer dan perangkat lunak. Beberapa jenis server yang memberikan yang mendukung layanan web antara lain: Web Server, Transaction Server, Proxy Server, Cache Server, Wireless Gateway Server, dan Mirror Server.

- Dewasa ini pemakaian web server semakin penting bagi dunia bisnis, karena semakin banyak informasi dan layanan-layanan sebuah perusahaan diinformasikan melalui web. Semakin banyak informasi diberikan melalui sebuah web, permintaan layanan terhadap web tersebut semakin banyak. Semakin banyak permintaan terhadap web, maka semakin tinggi probabilitas bahwa users menunggu respon dalam waktu lama. Akibatnya, pelanggan akan menjadi frustasi dan berpindah ke situs yang lain.

- Masalah unjuk kerja web server pada internet juga dipengaruhi oleh sifat

permintaan informasi dan layanan yang tidak dapat diprediksi. Misalnya pada waktu-waktu tertentu web server dalam kondisi idle, tidak ada pengunjung sama sekali. Tetapi dapat terjadi pada waktu yang lain tiba-tiba lalu-lintas informasi meningkat dengan cepat.

- Disamping jumlah pengunjung, hal lain yang mempengaruhi unjuk kerja sebuah web server adalah ukuran dari obyek (misalnya: teks, grafik, video, audio) yang diambil dari web server bervariasi dari 10^3 sampai 10^7 byte.

Satuan Ukur

Note:

Waktu response dan throughput merupakan dua satuan ukur penting untuk sistem web. Laju layanan permintaan HTTP adalah satuan ukur umum yang digunakan untuk menghitung throughput dari sistem web dan mungkin diekspresikan sebagai operasi HTTP per detik. Karena bervariasinya obyek yang dapat diberikan oleh sebuah sistem web, throughput biasanya juga diukur dengan menggunakan satuan bit per second (bps).

Satuan ukur lain yang digunakan untuk menghitung aktifitas web adalah *hit*, yang berarti semua koneksi ke sebuah situs web termasuk permintaan dan error. Pada sebuah situs, sebuah halaman dapat mengandung berbagai macam file tergantung pada kompleksitas dari situs web tersebut. Karena itu hit tidak lagi digunakan sebagai satuan ukur unjuk kerja dikarenakan sulitnya proses perbandingan antara satu situs dengan situs yang lain.

Beberapa permintaan oleh seorang user pada sebuah situs web disebut sebagai *visit*. Sementara itu beberapa permintaan selama visit disebut dengan *session*. Sebuah session atau visit dibedakan berdasarkan periode waktu time-out. Jika seorang user tidak melakukan aktifitas selama kunjungan sebuah situs dalam suatu waktu time-out tertentu, maka interaksi berikutnya dengan situs web memulai session baru.

Secara umum, satuan ukur unjuk kerja layanan web yang sering digunakan adalah sebagai berikut:

- end-to-end response time
- site response time
- throughput dalam permintaan/sec
- throughput dalam Mbps
- error per second
- pengunjung perhari

Latihan 1

Sebuah situs dari sebuah travel agent dimonitor selama 30 menit dan terdapat sebanyak 9000 permintaan HTTP dihitung. Server web dapat memberikan tiga macam obyek layanan, yaitu halaman HTML, gambar dan video. Melalui observasi diketahui bahwa dokumen HTML direpresentasikan oleh 30% dari permintaan dengan ukuran rata-rata 11.200 byte. Sedangkan gambar terhitung 65% dari permintaan dengan ukuran rata-rata 17.200 byte. Video hanya 5% dari seluruh permintaan dengan ukuran rata-rata 439.000 byte. Berapa throughput dari server tersebut?

Sumber dari Delay

Note:

Salah satu tugas utama dalam menentukan unjuk kerja dari sebuah web server adalah mengetahui sumber dari delay. Seperti terlihat dalam Gambar 8.1 sumber delay dapat dikategorikan dalam 4 hal, yaitu: (1) Fase pencarian DNS, (2) Fase koneksi TCP, (3) Waktu eksekusi oleh server dan (4) Waktu transmisi melalui jaringan internet atau intranet. Langkah pertama dari eksekusi Web server adalah pencarian DNS yang mana browser mengkonversi nama yang diberikan dalam bentuk URL ke nomor IP guna menetapkan koneksi TCP

pada fase berikutnya. Secara rata-rata proses pencarian DNS membutuhkan waktu 0.01 sampai 0.11 sec untuk situs-situs e-commerce dunia. Tiga macam delay yang lain dijelaskan pada uraian berikut.

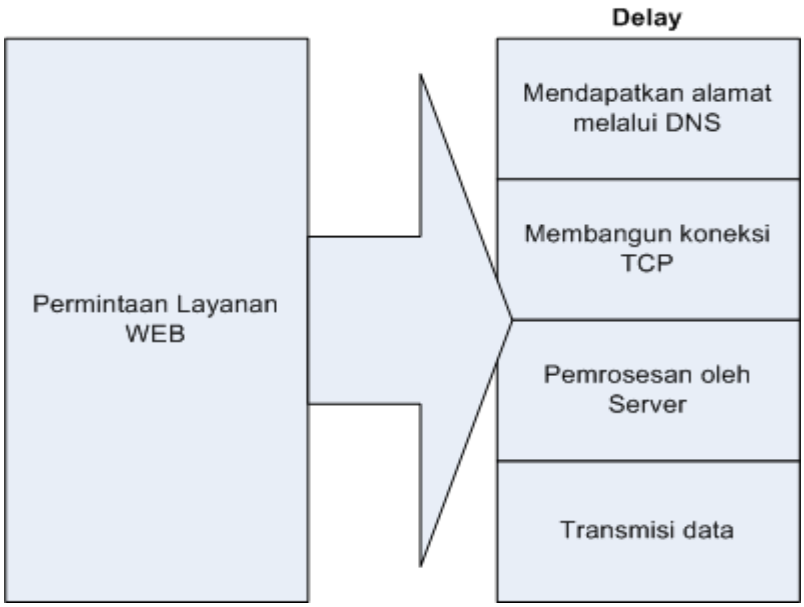

Figure 8.1: Komponen dari delay pada layanan web.

Anatomi dari Transaksi WEB

Dalam sistem terdistribusi seperti pada WWW, apabila suatu kesalahan terjadi pada salah satu komponen dari sistem maka sangat sulit melakukan penelusuran terhadap letak kesalahan tersebut. Karena itu sangatlah penting untuk memahami proses transaksi dari sistem web secara keseluruhan. Transaksi web dapat dibagi dalam 3 komponen dari sistem web, yaitu browser, network dan server seperti terlihat dalam Gambar 8.2.

1. **Browser**.
   - User melakukan klik pada sebuah hyperlink dan meminta dokumen.
   - Browser pada komputer lokal mencari alamat tersebut pada cache dan mengembalikan dokumen kepada browser. Dalam hal ini waktu response dari user dinotasikan oleh $R'_{\text{Browser,hit}}$.

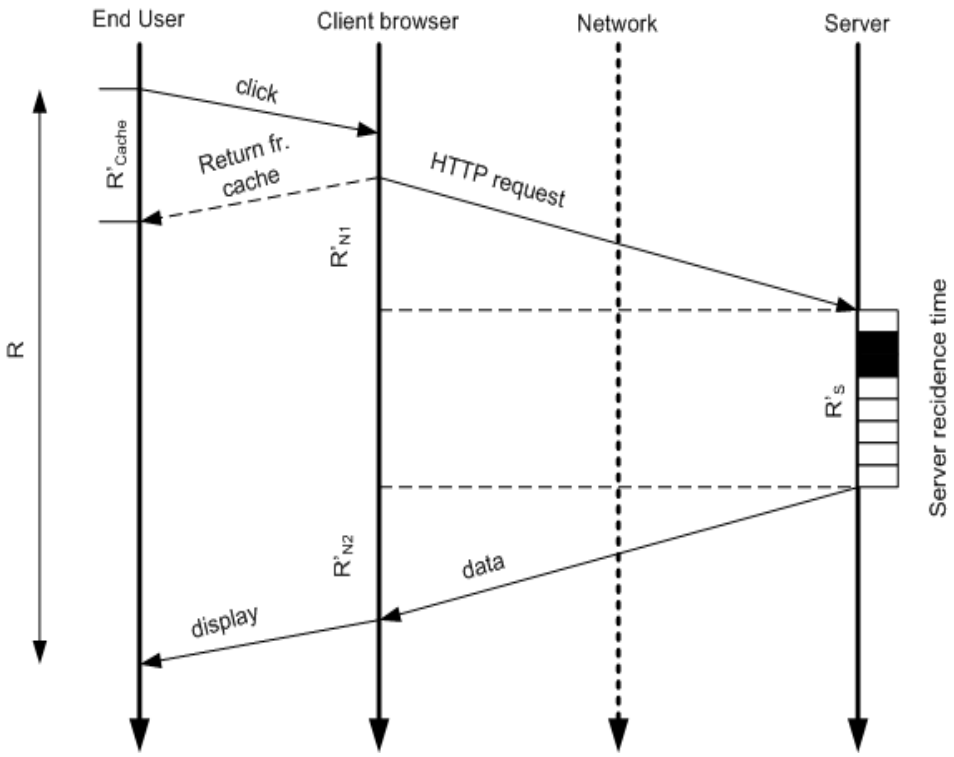

Figure 8.2: Anatomi dari transaksi web.

- Apabila tidak ditemui pada cache (miss), maka:
  - browser bertanya kepada DNS untuk melakukan konversi nama server ke sebuah nomor IP.
  - client membuka koneksi TCP dengan server.
  - client mengirimkan HTTP request ke server.
- Selama menerima response dari server, browser melakukan pemformatan dan penampilan dokumen beserta image. Dokumen tersebut disimpan dalam cache. Waktu pemrosesan ini dinotasikan sebagai $R'_{\text{Browser,miss}}$.

2. **Network**.

- Network memberikan kontribusi delay pada proses pengiriman informasi dari client ke server, disimbolkan oleh R'_{N1} pada gambar dan pengiriman dari server ke client, disimbolkan olehang R'_{N2}. Delay

ini merupakan komponen dari berbagai macam rute yang dilalui oleh data, misalnya client, server, modem, router, link komunikasi, switch dan sebagainya. Total waktu dari HTTP request dalam jaringan disimbolkan oleh R'_{Network}, dimana $R'_{\text{Network}} = R'_{N1} + R'_{N2}$.

3. **Server**.

   - Permintaan datang dari server.
   - Server menterjemahkan permintaan.
   - Server melakukan eksekusi terhadap method of request (misalnya GET, HEAD, dsb.).
   - Apabila permintaan yang dimaksud adalah GET, server mencari dokumen tersebut di dalam cache atau disk.
   - Server mengambil file dokumen tersebut dari cache atau disk dan menuliskannya pada network port.
   - Pada saat semua file telah terkirim, maka server melakukan penutupan koneksi.
   - Waktu yang dibutuhkan untuk eksekusi HTTP request disimbolkan sebagai R'_{server}.

Apabila dokumen yang diminta tidak ditemukan pada cache dari client, maka waktu total dari permintaan adalah:

$$R_{\text{miss}} = R'_{\text{Browser,miss}} + R'_{\text{Network}} + R'_{\text{server}}. \tag{8.1}$$

Apabila dokumen yang diminta ditemukan dalam cache, maka waktu response yang dibutuhkan adalah:

$$R_{\text{hit}} = R'_{\text{Browser,hit}}. \tag{8.2}$$

Biasanya $R_{\text{hit}} \lll R_{\text{miss}}$. Misalkan kita anggap bahwa browser menemukan data yang diminta pada cache lokal sebanyak N_C kali dari setiap N_T per-

mintaan. Maka waktu response rata-rata, R dari N_T permintaan dapat dituliskan sebagai:

$$R = p_C \times R_{\text{hit}} + (1 - p_C) \times R_{\text{miss}}, \tag{8.3}$$

dimana $p_C = N_C/N_T$ adalah probabilitas bahwa data ditemukan pada cache.

Latihan 2

Seorang pengguna ingin melakukan analisa terhadap pengaruh ukuran cache lokal pada waktu response dari web. Ia mengamati bahwa 20% dari permintaan dilayani oleh cache lokal dengan waktu response rata-rata 400 msec. Jika waktu response total untuk meminta dokumen pada web server adalah 3 s, berapa response waktu permintaan yang dialami oleh user? Apabila diketahui bahwa peningkatan ukuran dari cache dapat meningkatan probabilitas data ditemukan pada cache sebanyak 45%, berapa waktu response yang dialami oleh user sekarang?

BottleNeck

Pada saat jumlah client dan server bertambah, unjuk kerja sistem web yang dirasakan oleh client dipengaruhi juga oleh unjuk kerja berbagai perangkat dan sistem yang terletak diantara client dan server. Salah satu komponen yang membatasi unjuk kerja dari sistem adalah adanya bottleneck.

Latihan 3

Seorang pengguna internet mengeluh akibat lambatnya proses download dokumen-dokumen HTML yang berukuran menengah (rata-rata berukuran 20 KB). Untuk mengatasi masalah tersebut, user berkeinginan untuk melakukan upgrade processor dengan kecepatan dua kali lipat. Namun sebelum membelanjakan uang, ia ingin menjawab pertanyaan "Bagaimana peningkatan waktu response apabila kecepatan prosesor ditingkatkan?" Anggaplah waktu response jaringan (R'network) untuk membawa dokumen adalah 7.5 sec. Waktu dari server mem-

proses data adalah 3.6 sec dan waktu yang dibutuhkan oleh browser untuk memproses data adalah 0.3 sec. Hitung waktu response total yang dibutuhkan sebelum dan sesudah upgrade!

Latihan 4

Sebuah perusahaan obat-obatan bermaksud menggunakan intranet untuk menyebarkan gambar-gambar struktur molekul dari bahan-bahan kimia yang dibuat oleh perusahaan tersebut. Intranet ini akan digunakan untuk training secara online. Setiap kelas terdiri atas 100 karyawan, dan kita dapat menganggap secara rata-rata terdapat 80% dari siswa sedang aktif pada waktu tertentu. Selama kelas berlangsung, setiap siswa rata-rata melakukan akses sebanyak 100 kali per jam. Setiap permintaan akses tersebut rata-rata terdapat 5 gambar dengan ukuran rata-rata 25.600 byte. Berapa bandwidth minimum yang dibutuhkan untuk koneksi ke server tersebut?

Service Time pada Network

Pesan apapun yang dikirimkan oleh client kepada server akan melewati beberapa lapisan protocol dan juga melintasi beberapa Network. Sebagai contoh seperti terpampang dalam Gambar 8.3, pesan yang akan dikirimkan ke Server harus melewati tiga buah Local Area Network, yaitu Jaringan Ethernet 10 Mbps, jaringan FDDI Ring 100 Mbps dan jaringan Tokon Ring 16 Mbps.

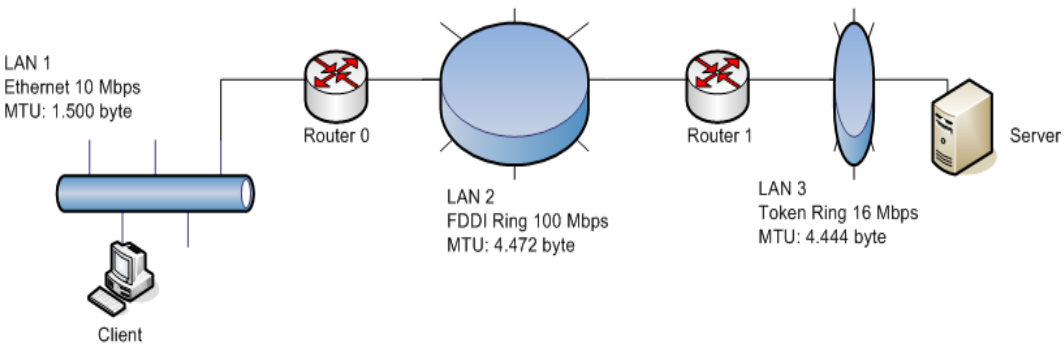

Figure 8.3: Koneksi dari Client menuju ke Server.

Selain itu, pesan juga akan melewati beberapa layer protokol, antara lain: protokol pada lapisan Transport, protokol pada lapisan Internet dan protokol pada lapisan Network Access dimana masing-masing lapis akan menambahkan header sesuai dengan kebutuhan masing-masing lapis. Header dan Data akan membentuk PDU (Protocol Data Unit). Pada lapis Internet, ukuran data dibatasi oleh MTU dari masing-masing perangkat jaringan. Seperti ditunjukkan dalam Gambar 8.3 MTU untuk jaringan ethernet adalah 1.500 byte, MTU untuk jaringan FDDI ring adalah adalah 4.472 byte dan MTU untuk jaringan Token Ring adalah 4.444 bytes. Karena itu Router akan selalu melakukan proses Fragmentasi dan Reassembly fragment. Spesifikasi protokol dan MTU masing-masing ditunjukkan dalam Tabel 8.1.

Table 8.1: Karakteristik Protokol

Protokol	Nama PDU	Ukuran PDU Max. (byte)	Overhead (byte)	Ukuran Data Max. (byte)
TCP	Segment	*	20	*
UDP	Datagram	*	8	*
IP v.4	Datagram	65.535	20	65.515
IP v.6	Datagram	65.535	40	65.495
ATM	Cell	53	5	48
Ethernet	Frame	1.518	18	1.500
IEEE 802.3	Frame	1.518	21	1.497
IEEE 802.5	Frame	4.472	28	4.444
FDDI	Frame	4.500	28	4.472
*: dibatasi oleh ukuran dari IP Datagram.				

Service time adalah waktu yang dibutuhkan untuk mentransmisikan pesan melewati network. Waktu ini merupakan perbandingan antara jumlah byte yang dibutuhkan untuk mentransmisikan pesan (termasuk header) dibagi den-

gan bandwidth dari Network.

Latihan 5

Sebuah client mengirimkan 300 byte permintaan kepada web server dan menerima 10.000 byte balasan. Interaksi antara client dan server ini terjadi pada koneksi TCP. Data permintaan ini diletakkan pada segment TCP yang selanjutnya melintasi jaringan dalam bentuk IP Datagram. Pada masing-masing LAN (seperti terlihat dalam Gambar 8.3) IP Datagram dibungkus oleh frame Ethernet, frame FDDI dan frame Token Ring. Pada data permintaan tersebut akan ditambahkan header TCP 20 byte dan header IP sebanyak 20 byte. Masing-masing LAN juga akan menambahkan frame: 18 byte untuk LAN Ethernet, 28 byte untuk LAN FDDI Ring dan 28 byte untuk LAN Token Ring. Tentukan waktu yang dibutuhkan untuk mentransimisikan pesan pada masing-masing LAN tersebut!

Asumsikan bahwa pada saat koneksi TCP terjadi Maximum Segment Size (MSS) yang digunakan adalah 1.460 bytes. Karena itu Server harus mengirimkan sebanyak 7 buah segment agar 10.000 byte data dapat dikirimkan. Enam segment pertama memiliki ukuran 1.460 byte ditambah dengan header TCP sebanyak 20 byte. Segment terakhir adalah 1.240 byte ($10.000 - 6 \times 1.460$). Hitunglah waktu yang dibutuhkan untuk mentransimisikan pesan dari server ke client pada masing-masing LAN tersebut!

Latihan 6

Dengan mengacu pada Gambar 8.3, client melakukan permintaan kepada Web Server dengan laju 3 permintaan per menit atau 0,05 permintaan per detik. Ukuran rata-rata dari pesan permintaan adalah 400 byte. 80% pesan balasan dari Web Server berukuran rata-rata 8.092 byte dan 20% pesan balasan berukuran rata-rata 100.000 byte. Asumsikan tidak ada proses fragmentasi.

1. Berapa waktu layan (Service Time) dari setiap pesan permintaan dan

pesan balasan untuk setiap LAN dengan mengasumsikan bahwa MSS adalah 1.460 byte!

2. Dengan menggunakan hasil perhitungan di atas, tentukan utilisasi jaringan untuk setiap LAN dengan jumlah client sebagai berikut: 40, 80, 120, 160, 200, 240, 280!

Note:

- Utilisasi jaringan rata-rata didapatkan dari perkalian antara laju kedatangan rata-rata dengan waktu layan rata-rata pesan permintaan.
- Waktu layan rata-rata untuk setiap permintaan pada sembarang jaringan sama dengan waktu layan rata-rata permintaan ditambah dengan waktu layan rata-rata dari pesan balasan, dimana pesan balasan terdiri atas: 80% dari pesan balasan pendek dan 20% pesan balasan panjang.
- Waktu kedatangan total dari permintaan pada setiap LAN adalah sama dengan jumlah client dikalikan dengan laju kedatangan rata-rata setiap permintaan untuk setiap client.

Secara umum rumusan-rumusan matematis yang terlibat dalam perhitungan waktu layan suatu network adalah:

Jumlah datagram yang dihasilkan olehn suatu pesan (tanpa ada proses fragmentasi):

$$\text{NDatagram} = \left\lceil \frac{\text{MessageSize}}{\text{MSS}} \right\rceil. \tag{8.4}$$

Total overhead protokol dari pesan yang melintasi network n adalah:

$$\text{Overhead}_n = \text{NDatagrams} \times (\text{TCPOvd} + \text{IPOvd} + \text{FrameOvd}_n). \tag{8.5}$$

Waktu layan dari pesan pada network n adalah:

$$\text{ServiceTime}_n = \frac{(\text{MessageSize} + \text{Overhead}_n) \times 8}{10^6 \times \text{Bandwidth}_n}. \tag{8.6}$$

Utilisasi dari network n dapat dihitung dengan rumusan:

$$U_n = \sum_{\text{pesan ke-}j} \lambda_j \times \text{ServiceTime}_n^j, \tag{8.7}$$

dimana λ_j adalah laju kedatangan dari pesan ke j, sedangkan ServiceTime_n^j adalah wakan layan rata-rata dari pesan ke-j pada network n.

Service Time pada Router

Router merupakan sebuah perangkat jaringan untuk memproses rute dari paket data yang melewatinya. Datagram yang masuk ke dalam router akan diantrikan terlebih dahulu sampai prosesor dari router siap melayani paket data tersebut. Selanjutnya router akan melihat alamat tujuan dari datagram dan membandingkan dengan tabel routing yang dimiliki. Datagram yang telah diproses akan dikirimkan ke antrian output seperti terlihat dalam Gambar 8.4.

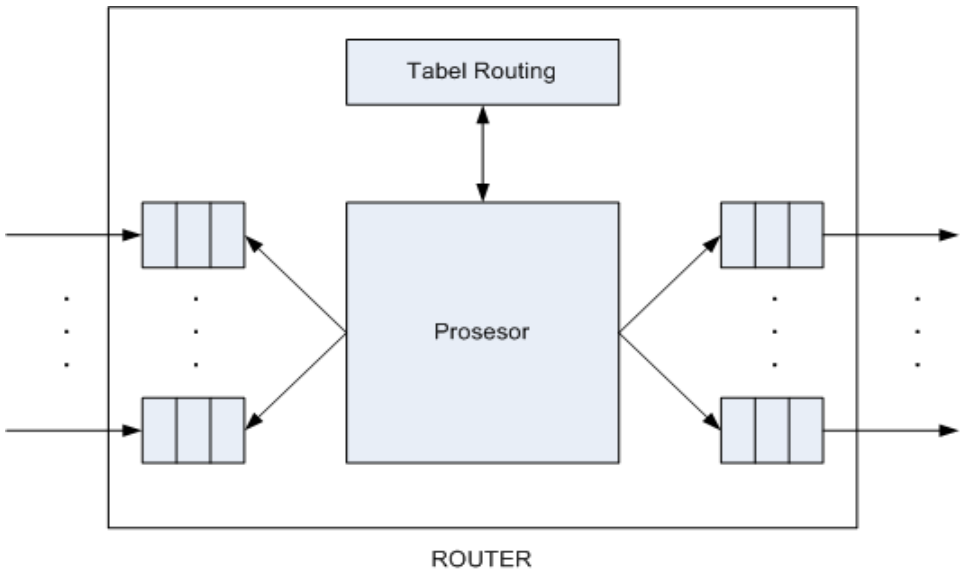

Figure 8.4: Antrian pada Router.

Waktu yang dibutuhkan oleh router untuk memproses datagram disebut sebagai *router latency*. Biasanya ukuran dari router latency ini diberikan oleh vendor pembuat router dalam hitungan mili detik. Berdasarkan router latency, secara matematis waktu layan dari router dapat dirumuskan sebagai:

$$\text{RouterServiceTime} = \text{NDatagrams} \times \text{Routerlatency}. \tag{8.8}$$

Latihan 7

Pada contoh kasus Latihan 6, asumsikan bahwa Router 1 dan ROuter 2 dapat memproses paket dengan laju 400.000 paket/sec, maka tentukan: Waktu layan per paket data dari router dan tentukan waktu layan dari router tersebut untuk memproses permintaan, balasan pendek dan balasan panjang!

Waktu Download dari Halaman Web

Sebagaimana diketahui bahwa sebuah halaman web dapat terdiri atas berbagai macam obyek meliputi halaman HTML dan obyek-obyek lain, seperti gambar, script, audio, video dsb. Karena itu bagian akan merumuskan secara matematis waktu yang dibutuhkan untuk mendownload halaman web. Asumsi dalam perhitungan ini adalah:

- Waktu yang dibutuhkan oleh browser untuk menampilkan data tidak dihitung.
- Waktu yang dibutuhkan oleh server untuk memproses data tidak dihitung.
- Tidak lebih dari satu buah web server terlibat (Tidak ada redirection).
- Round Trip Time (RTT) dan bandwidth efektif untuk setiap halaman HTML beserta obyek-obyek dari web adalah sama.

Beberapa notasi yang akan digunakan dalam perhitungan ini adalah:

- HTTPHeader: jumlah header pada HTTP response dalam byte (sekitar 290).
- NOby: jumlah obyek dalam pada halaman web.

- O_i: ukuran dalam byte dari obyek ke-i, dimana $i = 0, \cdots, \text{NOby}$; Obyek ke-0 adalah halaman HTML itu sendiri.
- RTT (Round Trip Time) : waktu yang dibutuhkan untuk berjalan dari client menuju ke server dan dari server kembali menuju ke client dalam hitungan detik.
- MSS: Maximum Segment Size dalam byte.
- B: Bandwidth efektif antara browser dan web server dalam satuan byte/detik.

Terdapat dua macam cara untuk melakukan koneksi HTTP, yaitu melalui *non-persistent connection* sebagaimana dilakukan oleh HTTP/1.0 dan model yang kedua adalah dengan menggunakan *persistent connection* sebagaimana dilakukan oleh HTTP/1.1. Pada non-persistent connection, HTTP server akan membuka port setelah koneksi TCP ditetapkan dan segera menutup kembali setelah sebuah obyek terambil. Untuk mengambil obyek berikutnya, maka dibutuhkan proses untuk penetapan koneksi TCP kembali dan seterusnya. Sedangkan pada persistent connection, koneksi TCP akan bertahan sampai waktu tertentu. Karena itu pada persistent connection beberapa obyek dapat diambil pada waktu besamaan sampai waktu penutupan dilakukan oleh server. Perhatikan Gambar 8.5.

Ukuran sebuah halaman web total termasuk header dari HTTP dalam ukuran byte adalah:

$$\text{PageSize} = \sum_{i=0}^{\text{NOby}} (O_i + \text{HTTPHeader}). \tag{8.9}$$

Jumlah paket yang dibutuhkan untuk mentransfer halaman adalah:

$$\text{Npackets} = \left\lceil \frac{\text{PageSize}}{\text{MSS}} \right\rceil \tag{8.10}$$

Pada non-persistent connection, dibutuhkan dua kali RTT sebelum byte pertama dari setiap dokumen tiba. RTT pertama dibutuhkan untuk melakukan penetapan koneksi TCP. RTT kedua dibutuhkan untuk mengirimkan HTTP

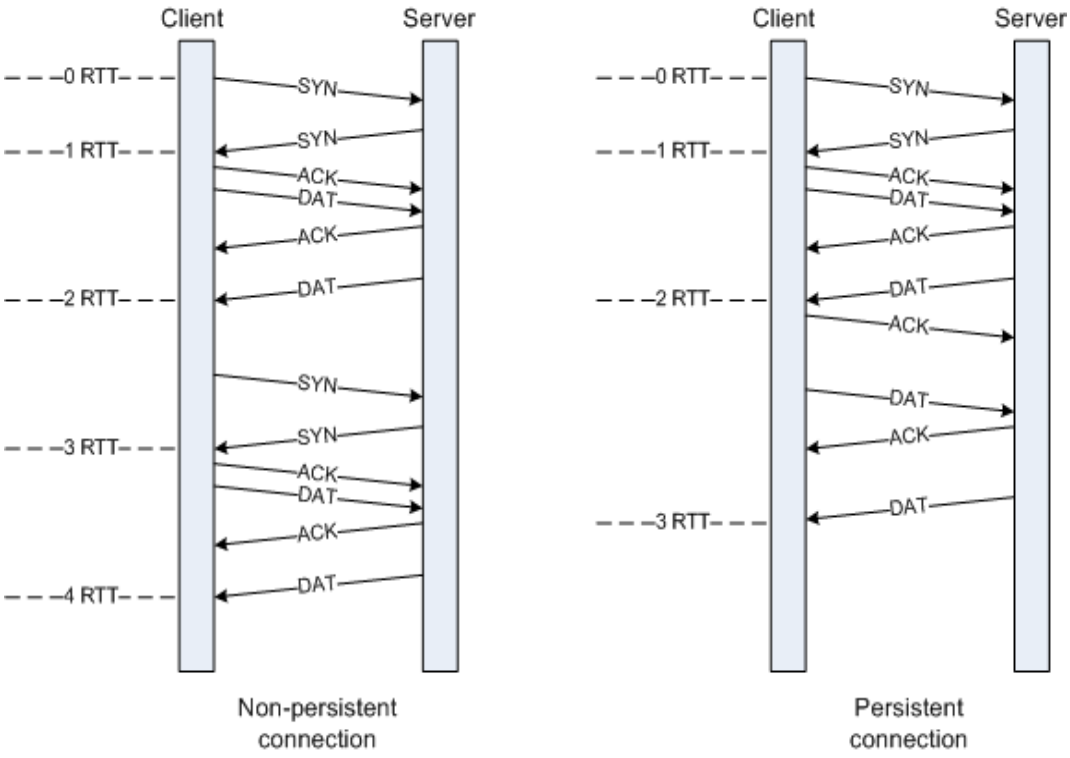

Figure 8.5: non-persistent dan persistent connection pada HTTP.

request bersama-sama dengan konfirmasi ACK dari client kepada server dan HTTP response dari server kepada client. Karena itu waktu download halaman (page download time) web minimal pada non-persistent connection adalah:

$$\mathrm{PDT_{NP}} < (\mathrm{NOby} + 1)(2 \times \mathrm{RTT}) + \frac{\mathrm{PageSize}}{B}. \tag{8.11}$$

Untuk persistent connection, hanya dibutuhkan satu RTT untuk penetapan koneksi TCP dan satu RTT untuk setiap download halaman HTML dan masing-masing obyek. Waktu download halaman web minimal pada persistent connection dirumuskan oleh persamaan:

$$\mathrm{PDT_P} < \mathrm{RTT}) + (\mathrm{NOby} + 1)\mathrm{RTT}) + \frac{\mathrm{PageSize}}{B}. \tag{8.12}$$

Latihan 8

Seorang designer web akan melakukan evaluasi terhadap waktu download dengan menggunakan dua buah alternatif. Desain pertama menggunakan halaman

HTML dengan 15.650 byte dan 10 gambar yang masing-masing memiliki ukuran 4.200 byte. Desain kedua dengan ukuran file lebih besar, yaitu halaman HTML dengan ukuran sama tetapi dengan 20 gambar yang masing-masing memiliki ukuran 20.000 byte. Hitunglah waktu download halaman minimal untuk kasus non-persistent connection dan persistent connection dengan asumsi bahwa RTT=0.05 sec, MSS = 1.460 byte dan B = 125 KB/sec. Hitung kembali waktu download halaman tersebut untuk B = 56 Kbps. Kesimpulan apa yang dapat ditarik dari pengaruh waktu download terhadap penurunan bandwidth?

Seperti terlihat dalam rumusan matematis di atas, waktu download minimal tidak memperhatikan pengaruh dari *Window Size* protokol TCP. Apabila komponen ini diperhitungkan, maka rumusan waktu download halaman web untuk non-persistent connection adalah:

$$\begin{aligned} \mathrm{PDT_{NP}} &= (\mathrm{NOby}+1)\times 2\times \mathrm{RTT} + \frac{\mathrm{PageSize}}{B} \\ &\quad \sum_{i=0}^{\mathrm{NOby}} [P_i(\mathrm{RTT}+\mathrm{MSS}/B) - (2^{P_i}-1)\mathrm{MSS}/B]. \end{aligned} \tag{8.13}$$

dimana

$$P_i = \min\left\{ \lceil \log_2(1+\frac{\mathrm{RTT}}{\mathrm{MSS}/B})\rceil + 1, \lceil \log_2(\frac{O_i}{\mathrm{MSS}}+1)\rceil - 1 \right\}. \tag{8.14}$$

Sedangkan untuk persistent connection, rumusan waktu download halaman adalah:

$$\begin{aligned} \mathrm{PDT_P} &= \mathrm{RTT} + \frac{\mathrm{PageSize}}{B} + (\mathrm{NOby}+1)\times \mathrm{RTT} + \\ &\quad Q(\mathrm{RTT}+\mathrm{MSS}/B) - (2^Q-1)\mathrm{MSS}/B. \end{aligned} \tag{8.15}$$

dimana

$$Q = \min\left\{ \lceil \log_2(1+\frac{\mathrm{RTT}}{\mathrm{MSS}/B})\rceil + 1, \lceil \log_2(\frac{\mathrm{PageSize}}{\mathrm{MSS}}+1)\rceil - 1 \right\}. \tag{8.16}$$

Latihan 9

Sebuah halaman HTML dari sebuah situs web memiliki ukuran 44.087 byte

dan 17 gambar di dalamnya dengan ukuran masing-masing adalah: 19.288, 4.243, 5.362, 3.030, 3.011, 265, 654, 586, 608, 3.913, 7.303, 6.052, 1.195, 640, 1.470, 2.538 dan 2.984. Dengan mengasumsikan bahwa bandwidth yang dapat digunakan adalah 10Mbps, ukuran segment 1.460 byte dan RTT = 0.05 sec, hitunglah waktu download halaman untuk non-persistent dan persistent connection.

Appendix A

Dasar Statistik dan Stokastik

Menyimpulkan data dengan sebuah angka.

Note:

- Sebuah angka dapat digunakan untuk merepresentasikan karakteristik kunci dari satu set data. Misalnya, rata-rata (averaging).
- Tiga alternatif yang digunakan untuk menyimpulkan satu data antara lain: mean, median dan mode. Ketiganya seringkali disebut sebagai: *indices of central tendencies*, karena ketiganya menentukan titik tengah dari distribusi sebuah sample data.
- Sample mean didapat dengan cara menjumlahkan semua data observasi dan membaginya dengan jumlah obersevasi.
- Sample median didapat dengan cara melakukan pengurutan data dari kecil ke besar dan mengambil nilai tengah dari deretan tersebut.
- Sample mode dilakukan dengan cara melakukan plotting histogram dan mengambil nilai tengah dari histogram yang memiliki nilai tertinggi.
- Mean dan median selaku ada dan unik, tetapi mode tidak selalu ada. Misalnya, pada sebuah observasi dimana seperangkat data menghasilkan nilai sama.

- Letak mean, median dan mode untuk berbagai macam distribusi ditunjukkan dalam Gambar A.1

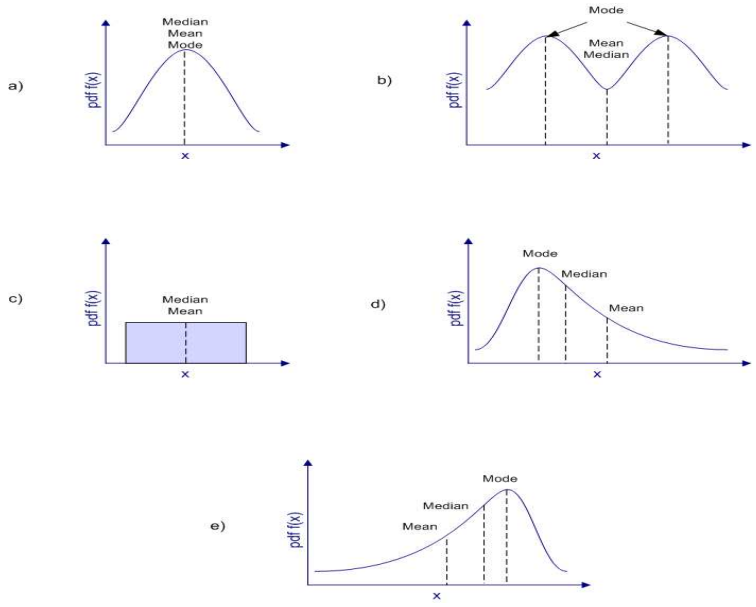

Figure A.1: Mean, median dan mode pada berbagai distribusi.

- Problem utama dengan mean adalah sangat dipengaruhi oleh outliers, sedangkan median dan mode tidak.
- Mean memberikan bobot yang sama untuk setiap data observasi.
- Mean memiliki properti penjumlahan dan linearitas dimana mean dari penjumlahan adalah penjumlahan dari rata-rata. Sifat ini tidak berlaku pada median dan mode.

Memilih diantara mean, median dan mode. *Note:*

Guideline untuk memilih ketiga centra tendency dapat dilihat pada flowchart pada Gambar A.2

Contoh:

- *Resource yang paling sering digunakan oleh sistem:* Resource adalah categorical data karena harus digunakan mode.

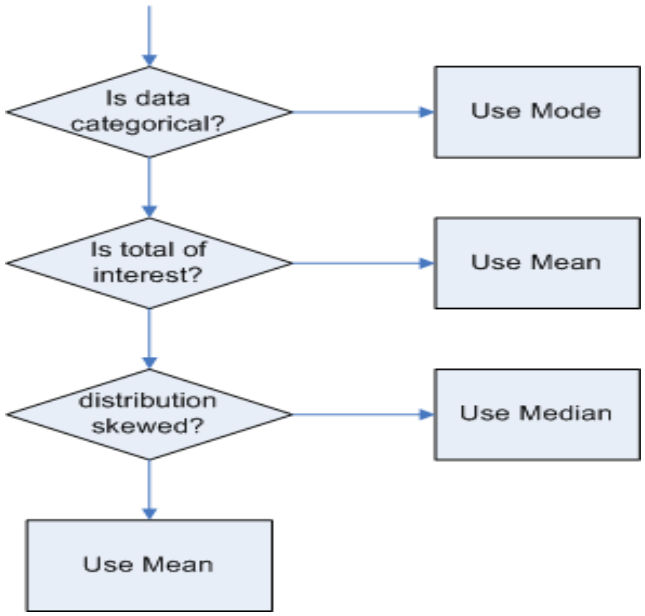

Figure A.2: Flowchart pemilihan mean, median dan mode.

- *Waktu antar-kedatangan (Interarrival Time):* karena kita tertarik pada keseluruhan waktu maka mean paling cocok digunakan.
- *Load pada computer:* median lebih disukai.

Kesalahan umum menggunakan mean. *Note:*

- Menggunakan mean untuk data-data yang berbeda secara signifikan. Misalnya, CPU time per query didapatkan sebesar 10 dan 100 ms. Maka menghitung mean sebesar 550 ms untuk kedua data diatas tidak berguna. Nilai mean sama sekali tidak mewakili kedua data tersebut. Dalam kondisi sepeti ini biasanya varian dapat digunakan melakukan uji kevalidan menggunakan mean.
- Menggunakan mean tanpa memperhatikan kemiringan (skewness) dari distribusi data. Sebagai contoh perhatikan Tabel A.1 di bawah ini:
- Mengalikan mean untuk mendapatkan mean dari hasil perkalian dari dua

Table A.1: Response Time sistem selama 5 hari

	Sistem A	Sistem B
	10	5
	9	5
	11	5
	10	4
	10	31
Sum	50	50
Mean	10	10
Mode	10	5

buah variabel random. Mean dari perkalian dua buah variabel random adalah sama dengan perkalian dari buah mean dari dua variabel random jika kedua variabel random tersebut saling bebas. Apabila keduanya saling berkorelasi, maka berlaku:

$$\mathrm{E}(xy) \neq \mathrm{E}(x)\mathrm{E}(y) \tag{A.1}$$

Geometric Mean.

Note:

Geometric Mean dari data sebanyak n yaitu $x_1, x_2, \ldots, x_n$ didapatkan dengan persamaan:

$$\dot{x} = \left(\prod_{i=1}^{n} x_i \right) \tag{A.2}$$

Geometric mean digunakan apabila kita tertarik pada hasil perkalian dari data observasi. Perhatikan contoh dibawah ini: Peningkatan unjuk kerja sebuah protocol baru pada ketujuh layer OSI ditunjukkan dalam Tabel A.2. Hitunglah rata-rata peningkatan pada setiap layer!

Table A.2: Peningkatan unjuk kerja pada 7-layer OSI

Layer	Performance Improvement (%)
7	18
6	13
5	11
4	8
3	10
2	28
1	5

Rata-rata peningkatan pada setiap layer

$$\begin{aligned} &= \{(1.18)(1.13)(1.11)(1.08)(1.10)(1.28)(1.05)\}^{1/7} - 1 \\ &= 0.13 \end{aligned}$$

Contoh-contoh kasus lain yang menggunakan geometric mean adalah:

- Cache hit ratio dari beberapa level cache
- Cache miss ratio
- Rata-rata error per hop pada jalur multihop dalam jaringan.

Contoh kasus: Jika IHSG meningkat 10% pada tahun pertama, 20% pada tahun kedua dan turun menjadi 15% pada tahun ketiga, berapa rata-rata kenaikan IHSG setiap tahun?

Harmonic Mean.

Note:

Harmonic mean dari sebuah sample $\{x_i, x_2, \ldots x_n\}$ didefinisikan sebagai:

$$\ddot{x} = \frac{n}{1/x_1 + 1/x_2 + \cdots + 1/x_n} \tag{A.3}$$

Harmonic mean dapat digunakan apabila aritmetik mean dapat dijustifikasi sebagai $1/x_i$. Misalnya, untuk menempuh jarak tertentu setengah perjalanan

kita bergerak dengan kelajuan 40 km/jam, pada setengah perjalanan yang lain kita tempuh dengan kelajuan 60 km/jam. Maka rata-rata kelajuan kita dihitung dengan harmonic mean menjadi 48 km/jam. Atau dengan kata lain, *total waktu* yang dibutuhkan untuk menempuh jarak tersebut akan sama jika kita bergerak dengan laju 48 km/jam untuk menempuh seluruh jarak tersebut.

Hal ini berlaku juga untuk menghitung rata-rata resistansi yang dokoneksikan secara paralel dari rangkaian listrik, apabila terdapat resistansi 40Ω dan 60Ω, maka rata-rata resistansi adalah 48Ω. Ini berarti, resistansi total yang diberikan akan sama apabila masing-masing resistor digantikan oleh resistor dengan resistansi 48Ω.

Contoh lain. Misalkan pengukuran berulang-ulang dilakukan untuk menghitung waktu yang dibutuhkan oleh sebuah benchmark pada sebuah prosesor. Anggaplah bahwa sebuah behcmark memiliki m juta instruksi, maka MIPS, x_i, dihitung dari repetisi ke-i adalah:

$$x_i = m/t_i \tag{A.4}$$

Sehingga, rata-rata MIPS dari prosesor tersebut adalah:

$$\begin{aligned} \ddot{x} &= \frac{n}{1/(m/t_1) + 1/(m/t_2) + \cdots + 1/(m/t_n)} \\ &= \frac{m}{(1/n)(t_1 + t_2 + \cdots + t_n)} \end{aligned} \tag{A.5}$$

Mean dari Rasio.

Note:

Masalah berikut yang sering muncul adalah menghitung rata-rata dari beberapa rasio. Beberapa aturan yang dapat digunakan adalah:

1. Jika kita mengambil jumlah dari pembilang dan penyebut dan keduanya berarti, maka rata-rata dari rasio adalah rasio dari rata-rata. Misalkan,

$x_i = a_i/b_i$, rata-rata dari rasio adalah:

$$\begin{aligned} \text{Average}\left(\frac{a_1}{b_1}, \frac{a_2}{b_2}, \cdots \frac{a_n}{b_n}\right) &= \frac{a_1 + a_2 + \cdots + a_n}{b_1 + b_2 + \cdots + b_n} \\ &= \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i} \\ &= \frac{(1/n)\sum_{i=1}^{n} a_i}{(1/n)\sum_{i=1}^{n} b_i} = \frac{\bar{a}}{\bar{b}}. \end{aligned} \tag{A.6}$$

Aplikasi yang sering menggunakan aturan ini adalah perhitungan rata-rata dari utilisasi. Sebagai contoh, utilisasi CPU daris ebuah sistem diukur lima kali dalam rentang waktu berbeda seperti ditunjukkan dalam Tabel A.3.

Table A.3: Utilisasi CPU

Measurement duration	CPU busy (%)
1	45
1	45
1	45
1	45
100	20
Sum	200
Mean	$\neq 40$

Mean dari utilisasi didapatkan dengan cara menghitung rasio dari total waktu sibuk CPU dan total waktu yang digunakan.

$$\begin{aligned} \text{MeanCPUutilization} &= \frac{0.45 + 0.45 + 0.45 + 0.45 + 20}{1 + 1 + 1 + 1 + 100} \\ &= 21\% \end{aligned}$$

2. Jika pembilang dan penyebut diharapkan mengikuti properti dari suatu perkalian, i.e. $a_i = cb_i$ dimana c adalah sebuah konstanta yang akan diaproksimasi, mana c dapat diestimasi dengan geometric mean

dari a_i/b_i. Sebagai contoh perhatikan Tabel A.4 yang menunjukkan beberapa benchmark yang dijalankan melalui program optimasi. Ukuran code sebelum dan sesudah dioptimasi diperlihatkan pada tabel beserta rasionya.

Table A.4: Optimasi Program Benchmark

	Code Size		
Program	Before	After	Ratio
BubbleP	119	89	0.75
IntmmP	158	134	0.85
PermP	142	121	0.85
PuzzleP	8612	7579	0.88
QueenP	7133	7062	0.99
QuickP	184	112	0.61
SieveP	2908	2879	0.99
TowersP	433	307	0.71
Geometric mean			0.82

Appendix B

Tabel Quantile dari Distribusi Normal.

A.2 QUANTITIES OF THE UNIT NORMAL DISTRIBUTION

Table A.2 lists z_p for a given p. For example, for a two-sided confidence interval at 95%, $\alpha = 0.05$ and $p = 1 - \alpha/2 = 0.975$. The entry in the row labeled 0.97 and column labeled 0.005 gives $z_p = 1.960$.

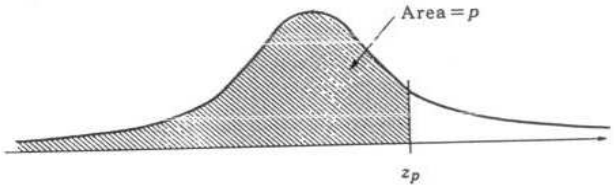

TABLE A.2 Quantiles of the Unit Normal Distribution

p	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.5	0.000	0.025	0.050	0.075	0.100	0.126	0.151	0.176	0.202	0.228
0.6	0.253	0.279	0.305	0.332	0.358	0.385	0.412	0.440	0.468	0.496
0.7	0.524	0.553	0.583	0.613	0.643	0.674	0.706	0.739	0.772	0.806
0.8	0.842	0.878	0.915	0.954	0.994	1.036	1.080	1.126	1.175	1.227

p	0.000	0.001	0.002	0.003	0.004	0.005	0.006	0.007	0.008	0.009
0.90	1.282	1.287	1.293	1.299	1.305	1.311	1.317	1.323	1.329	1.335
0.91	1.341	1.347	1.353	1.359	1.366	1.372	1.379	1.385	1.392	1.398
0.92	1.405	1.412	1.419	1.426	1.433	1.440	1.447	1.454	1.461	1.468
0.93	1.476	1.483	1.491	1.499	1.506	1.514	1.522	1.530	1.538	1.546
0.94	1.555	1.563	1.572	1.580	1.589	1.598	1.607	1.616	1.626	1.635
0.95	1.645	1.655	1.665	1.675	1.685	1.695	1.706	1.717	1.728	1.739
0.96	1.751	1.762	1.774	1.787	1.799	1.812	1.825	1.838	1.852	1.866
0.97	1.881	1.896	1.911	1.927	1.943	1.960	1.977	1.995	2.014	2.034
0.98	2.054	2.075	2.097	2.120	2.144	2.170	2.197	2.226	2.257	2.290

p	0.0000	0.0001	0.0002	0.0003	0.0004	0.0005	0.0006	0.0007	0.0008	0.0009
0.990	2.326	2.330	2.334	2.338	2.342	2.346	2.349	2.353	2.357	2.362
0.991	2.366	2.370	2.374	2.378	2.382	2.387	2.391	2.395	2.400	2.404
0.992	2.409	2.414	2.418	2.423	2.428	2.432	2.437	2.442	2.447	2.452
0.993	2.457	2.462	2.468	2.473	2.478	2.484	2.489	2.495	2.501	2.506
0.994	2.512	2.518	2.524	2.530	2.536	2.543	2.549	2.556	2.562	2.569
0.995	2.576	2.583	2.590	2.597	2.605	2.612	2.620	2.628	2.636	2.644
0.996	2.652	2.661	2.669	2.678	2.687	2.697	2.706	2.716	2.727	2.737
0.997	2.748	2.759	2.770	2.782	2.794	2.807	2.820	2.834	2.848	2.863
0.998	2.878	2.894	2.911	2.929	2.948	2.968	2.989	3.011	3.036	3.062
0.999	3.090	3.121	3.156	3.195	3.239	3.291	3.353	3.432	3.540	3.719

See Table A.3 for commonly used values.

Figure B.1: Tabel quantile dari distribusi normal.

Appendix C

Tabel Quantile dari Distribusi t.

A.4 QUANTILES OF THE t DISTRIBUTION

Table A.4 lists $t_{[p;n]}$. For example, the $t_{[0.95;13]}$ required for a two-sided 90% confidence interval of the mean of a sample of 14 observation is 1.771.

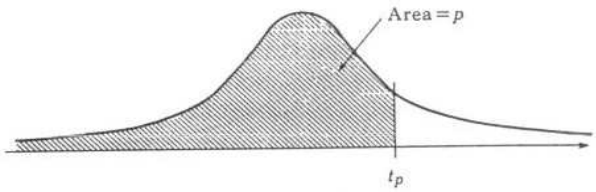

TABLE A.4 Quantiles of the t Distribution

	p							
n	0.6000	0.7000	0.8000	0.9000	0.9500	0.9750	0.9950	0.9995
1	0.325	0.727	1.377	3.078	6.314	12.706	63.657	636.619
2	0.289	0.617	1.061	1.886	2.920	4.303	9.925	31.599
3	0.277	0.584	0.978	1.638	2.353	3.182	5.841	12.924
4	0.271	0.569	0.941	1.533	2.132	2.776	4.604	8.610
5	0.267	0.559	0.920	1.476	2.015	2.571	4.032	6.869
6	0.265	0.553	0.906	1.440	1.943	2.447	3.707	5.959
7	0.263	0.549	0.896	1.415	1.895	2.365	3.499	5.408
8	0.262	0.546	0.889	1.397	1.860	2.306	3.355	5.041
9	0.261	0.543	0.883	1.383	1.833	2.262	3.250	4.781
10	0.260	0.542	0.879	1.372	1.812	2.228	3.169	4.587
11	0.260	0.540	0.876	1.363	1.796	2.201	3.106	4.437
12	0.259	0.539	0.873	1.356	1.782	2.179	3.055	4.318
13	0.259	0.538	0.870	1.350	1.771	2.160	3.012	4.221
14	0.258	0.537	0.868	1.345	1.761	2.145	2.977	4.140
15	0.258	0.536	0.866	1.341	1.753	2.131	2.947	4.073
16	0.258	0.535	0.865	1.337	1.746	2.120	2.921	4.015
17	0.257	0.534	0.863	1.333	1.740	2.110	2.898	3.965
18	0.257	0.534	0.862	1.330	1.734	2.101	2.878	3.922
19	0.257	0.533	0.861	1.328	1.729	2.093	2.861	3.883
20	0.257	0.533	0.860	1.325	1.725	2.086	2.845	3.850
21	0.257	0.532	0.859	1.323	1.721	2.080	2.831	3.819
22	0.256	0.532	0.858	1.321	1.717	2.074	2.819	3.792
23	0.256	0.532	0.858	1.319	1.714	2.069	2.807	3.768
24	0.256	0.531	0.857	1.318	1.711	2.064	2.797	3.745
25	0.256	0.531	0.856	1.316	1.708	2.060	2.787	3.725
26	0.256	0.531	0.856	1.315	1.706	2.056	2.779	3.707
27	0.256	0.531	0.855	1.314	1.703	2.052	2.771	3.690
28	0.256	0.530	0.855	1.313	1.701	2.048	2.763	3.674
29	0.256	0.530	0.854	1.311	1.699	2.045	2.756	3.659
30	0.256	0.530	0.854	1.310	1.697	2.042	2.750	3.646
60	0.254	0.527	0.848	1.296	1.671	2.000	2.660	3.460
90	0.254	0.526	0.846	1.291	1.662	1.987	2.632	3.402
120	0.254	0.526	0.845	1.289	1.658	1.980	2.617	3.373

Figure C.1: Tabel quantile dari distribusi t.