# Bab 3: Benchmarking pada Sistem Komputer

Dr. Ir. Yeffry Handoko Putra, M.T





## Benchmark

- Definisi Benchmarking:
  Serial tes dengan metriks dan workload yang telah dirancang sebelumnya
- Definisi Benchmark:
  Workload yang digunakan pada pengukuran Benchmarking, biasanya dirancang untuk memandingkan dua sistem yaitu yang berjenis **test workload**
- **Test Workload yang digunakan untuk membandingkan sistem komputer**
  1. Addition instruction
  2. Instruction mixes
  3. Kernels
  4. Synthetic programs
  5. Application benchmarks



## Addition Instruction (Early day of Computing)

- Jenis Workload yang pertama kali digunakan untuk membandingkan dua komputer
- Instruksi pada komputer mula-mula masih sedikit
- Perbandingan kinerja hanya didasari oleh komputer mana yang bisa mengerjakan instruksi penjumlahan dengan cepat



## Instruction Mix (Gabungan Instruksi)

- Definisi Instruction Mix:
  beberapa instruksi yang diberikan pembobotan sesuai dengan jumlah keseringan (frekuensi) pengunaannya dalam sistem komputer.
- Faktor pembobotan ini kemudian digunakan untuk waktu rata-rata instruksi
- Sehingga jika diberikan waktu tertentu dapat dihitung waktu rata-rata instruksi nya
- Yang paling sering digunakan adalah Gibson Mix
  Gibson Mix (1959, Jack C. Gibson):
  - Digunakan pada IBM 704
  - Kecepatan procesor diukur dalam memory cycle time, addition time, average of addition and multiplication time
  - Gibson Mix mengembanngkan sampai waktu rata-ratadari 13 instruksi

## Gibson Instruction Mix

**TABLE 4.1 Gibson Instruction Mix**

| | |
|---|---|
| 1. Load and Store | 31.2 |
| 2. Fixed-Point Add and Subtract | 6.1 |
| 3. Compares | 3.8 |
| 4. Branches | 16.6 |
| 5. Floating Add and Subtract | 6.9 |
| 6. Floating Multiply | 3.8 |
| 7. Floating Divide | 1.5 |
| 8. Fixed-Point Multiply | 0.6 |
| 9. Fixed-Point Divide | 0.2 |
| 10. Shifting | 4.4 |
| 11. Logical, And, Or | 1.6 |
| 12. Instructions not using registers | 5.3 |
| 13. Indexing | 18.0 |
| | 100.0 |

- **Laju rata-rata procesor kemudian dihitung dari pembobotan dari rata-rata waktu instrusi dari ke-13 instruksi gabungan tersebut**



## Saat ini Instruction Mix tidak digunakan

- Saat ini komputer menyediakan banyak kelas instruksi yang kompleks yang waktu instruksi rata-ratanya tidak berkaitan dengan gabungan instruksi
- Intruction Mix hanya merepresentasikan kinerja prosesor saja tidak seluruh sistem komputer
- Sistem komputer modern, waktu instruksi tergantung pada:
  - Addresing modes
  - cache hit rates
  - pipeline efficiency,
  - interference from other devices during processor-memory access cycles
  - Parameter value. Misal : frequency of zeros as a parameter, the distribution of zero digits in a multiplier, the average number of positions of preshift in floating-point add, and the number of times a conditional branch is taken
- Inversi dari waktu rata-rata instruksi : MIPS (Millions of Instructions Per Second) or MFLOPS (Millions of Floating-Point Operations Per Second) rates for the processor



## Kernels

- Adanya beberapa fasilitas yg membuat waktu instruksi sangat bervariasi, seperti: pipeline, caching, dll
- Instruksi diperlakukan sebagai kumpulan instruksi /fungsi bukan sebagai individual yang disebut service
- Service tersering (the most frequent function) disebut Kernel. Sering disebut Processing Kernel karena kernel Inisiasi lebih dikonsentrasikan pada kinerja prosesor daripada I/O
- Kernel adalah generalisasi dari gabungan instruksi. Beberapa operasi yang memanfaatkan kernel: inversi matriks, Sieve, Puzzle, Tree Searching, Fungsi Ackermann, Sorting
- Digunakan untuk membandingkan arsitektur komputer
- Namun tetap kinerja Kernel tidak merefleksikan kinerja total sistem karena Peralatan I/O tidak termasuk



## Synthetic Progam

- Karena processing kernel tidak digunakan pada I/O devices, maka untuk kinerja I/O dikembangkan lup pelatih (exercieser loop) yang dsebut service call atau I/O request
- Melalui Service call dapat dihitung rata-rata CPU time (AVT=Average CPU Time) dan waktu akhir (elapsed time) dari service call.
- Exerciser loop ditulis dalam high-level languages (FORTRAN, Pascal)
- Exerciser loop pertama disebut synthetic program (Buchloz, 1969) yang berisi i/O request.
- Synthetic Program digunakan untuk mengatur jumlah request sekaligus mengukur OS service seperti : process creation, forking, memory allocation

# Synthetic Progam

```
      DIMENSION Record (500)
!Control parameters:
      Num_Computes=500          !Repeat count for computation
      Num_Reads=35              !Number of records read
      Num_Writes=40             !Number of records written
      Num_Iterations=1000       !Repeat count for the experiment
!Open files:
      OPEN(UNIT=1,NAME='In.dat',TYPE='Old',
     1FORM='Unformatted',ACCESS='Direct')
      OPEN(UNIT=2,NAME='Out.dat'.TYPE='New',
     1FORM='unformatted',ACCESS='Direct')
      CALL Get_Time(CPU1.Elapsed1)    !Record starting time

      DO 500 Iteration=1,Num_Iterations

!Perform a number of read I/Os
      DO 100 i-1,Num_Reads
      READ(1'1),Record
100   CONTINUE
!Do computation
      DO 200 j=1,Num_Computes
      DO 200 i-1,500
200   Record(i)=1 + i + i*i + i*i*i
!Perform a number of write I/Os
      DO 300 i=1,Num_Writes
      WRITE(2'i).Record
300   CONTINUE
500   CONTINUE
      CALL Get_Time(CPU2,Elapsed2) !Get ending time
!Close files:
      CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      CPU_Time=(CPU2-CPU1)/Num_Iterations
      Elapsed_Time=(Elapsed2-Elapsed1)/Num_Iterations
      TYPE *,'CPU time per iteration is ',CPU_Time
      TYPE *.'Elasped time per iteration is '.Elapsed_Time
      STOP
      END
```

**FIGURE 4.1** Synthetic workload generation program.



# Popular Benchmark

- **Sieve**
- **Ackermann's Function**
- **Whetstone**
- **LINPACK**
- **Dhrystone**
- **Lawrence Livermore Loops**
- **Debit-Credit Benchmark**
- **SPEC Benchmark Suite**



# Sieve

- Sieve Kernel digunakan untuk membandingkan microprosesor, PC dan High-level language
- Didasari oleh Eratosthenes's sieve algorithm yang digunakan untuk mencari bilangan prima yang lebih kecil dari bilangan n
- Algoritma ini secara manual menuliskan semua bilangan integer dari 1 sampai n lalu menghapus kelipatan dari k untuk k = 2, 3, .., √*n*
- *Contoh :*

**1.** Write down all numbers from 1 to 20. Mark all as prime:

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11]
[12] [13] [14] [15] [16] [17] [18] [19] [20]

**2.** Remove all multiples of 2 from the list of primes:

[1] [2] [3] 4 [5] 6 [7] 8 [9] 10 [11] 12 [13] 14 [15] 16 [17] 18 [19] 20

**3.** The next integer in the sequence is 3. Remove all multiples of 3:

[1] [2] [3] 4 [5] 6 [7] 8 9 10 [11] 12 [13] 14 15 16 [17] 18 [19] 20

**4.** The next integer in the sequence is 5, which is greater than the square root of 20. Hence, the remaining sequence consists of the desired prime numbers



A Pascal program to implement the sieve kernel is given in Figure 4.2.

```
PROGRAM Prime (OUTPUT);
CONST
  MaxNum = 8191; (* Lists all primes up to MaxNum *)
  NumIterations = 10; (* Repeats procedure NumIterations times *)
VAR
  IsPrime : ARRAY [1..MaxNum] OF BOOLEAN;
  i,k,Iteration : INTEGER; (* Loop indexes *)
  NumPrimes : INTEGER: (* Number of primes found *)
BEGIN
  WRITELN('Using Eratosthenes Sieve to find primes up to ', MaxNum);
  WRITELN('Repeating it ',NumIterations,' times.');
  FOR Iteration := 1 TO NumIterations DO
    BEGIN (* Initialize all numbers to be prime *)
      FOR i := 1 TO MaxNum DO
        IsPrime[i] := TRUE;
      i := 2;
      WHILE i*i <= MaxNum DO
        BEGIN
          IF IsPrime[i] THEN
            BEGIN (* Mark all multiples of i to be nonprime *)
              k := i + i;
              WHILE k <= MaxNum DO

                BEGIN
                  IsPrime[k] := FALSE;
                  k := k + i;
                END; (* of WHILE k *)
            END; (* of If IsPrims *)
          i :+ i + 1;
        END; (* of WHILE i*i *)
    NumPrimes := 0;
    FOR i := 1 TO MaxNum DO (* Count the number of primes *)
      IF IsPrims[i] THEN NumPrimes := NumPrimes + 1;
      WRITELN(NumPrimes. ' primes');
    END; (* of FOR Iterations *)
  (* The following can be added during debugging to list primes. *)
  (* FOR i := 1 TO MaxNum DO IF IsPrime[i] THEN WRITELN (i); *)
END.
```

**FIGURE 4.2** Pascal program to implement sieve workload.

## Ackermann's Function

- Kernel ini digunakan untuk menilai efisiensi dari mekanisme pemanggilan prosedur dalam ALGOL-like languages.
- Fungsi Ackermann terdiri dari dua parameter dan didefinisikan rekursif. Fungsi Ackermann (3,*n*) mengevaluasi nilai n dari 1 sampai 6
- Nilai dari fungsi Ackermann (3,*n*) adalah $2^{n+3} - 3$.
  Fungsi ini digunakan untuk memverifikasi benchmark
- Kemudian benchmark yang dihasilkan adalah :
  - The average execution time per call
  - the number of instructions executed per call
  - the amount of stack space required for each call
- Jumlah pemanggilan rekursif dari Ackermann (3,2) dinyatakan dengan (Wichmann ,1976) :

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

This expression is used to compute the execution time per call. For Ackermann (3,*n*), the maximum depth of the procedure calls is $2^{n+3} - 4$. Hence, the amount of stack space required doubles when $n$ is increased by 1.



## Fungsi Ackermann ditulis dalam SIMULA

```
BEGIN
  INTEGER n;                !Loop index;
  INTEGER j;                !Function value;
  INTEGER num_calls;        !Number of recursive calls;
  INTEGER k;                !Contains 2**(n+3);
  INTEGER k1;               !Contains 4**(n-1);
  REAL t1,t2;               !CPU time values;

  INTEGER PROCEDURE Ackermann(m,n); VALUE m,n; INTEGER m,n;
    Ackermann := IF m=0 THEN n+1
                 ELSE IF n=0 THEN Ackermann(m-1,1)
                      ELSE Ackermann(m-1,Ackermann(m,n-1));

!Main Program;
  k := 16; K1 := 1;          !Initialize k and k1 for n=1;
  FOR n := 1 STEP 1 UNTIL 6 DO
    BEGIN
      t1 := CPUTIME;                !Beginning CPU time;
      j := Ackermann (3,n);         !Compute the function;
      t2 := CPUTIME;                !Ending CPU time;
      IF j <> k-3 THEN OUTTEXT("Wrong Value");
      OUTTEXT("Net CPU Time for Ackermann (3,");
      OUTINT(n,1); OUTTEXT(") is");
      OUTREAL(t2-t1,7,15); OUTIMAGE;
      Num_calls := (512*k1-15*k+9*n+37)/3;
      OUTTEXT("CPU Time per call:");
      OUTREAL((t2-t1)/num_calls.7,15);
      OUTIMAGE;
      ki := 4*k1;               !Update k1 for the next n;
      k := 2*k;                 !Update k for the next n;
    END
END
```

**FIGURE 4.3** SIMULA program to implement Ackermann's function.



## Whetstone Benchmark

- Synthetic Benchmark untuk mengevaluasi kinerja komputer, mula-mula dikembangkan pada National Physical Lab., UK. 1972 dengan bahasa Algol 60.
- Whetstone Benchmark terdiri dari 11 modules
- Selain sebagai syntehtic mix, Whetstone digunakan sebagai floating point benchmark pada cache memory. Namun hanya bertujuan pada optimisasi compiler sehingga parameter input I/O tidak mempengaruhi kinerja yang diukur
- Kernel Exercises whetstone (processor feature):
  - Array addressing
  - Fixed and floating point arithmetic
  - Subroutine calls
  - Parameter passing
- Whetstone kernel telah diterjemahkan ke FORTRAN, PL/I, dll
- Whetstone benchmark diukur dalam KWIPS (kilo Whetstone Instruction per second) juga MWIPS (Million Whestone Instruction per second)



## LINPACK Benchmark

- Dikembangkan oleh Jack Dongarra (1983) dari Argonne National Laboratory.
- Benchmark LINPACK terdiri dari sejumlah program yang menyelesaikan persamaan linier rumit menggunakan UNPACK subroutine. UNPACK berisi penjumlahan floating point dan perkalian.
- Contoh subroutinenya yang menyita waktu (time consumed) adalah BLAS (Basic Linear Algebra Subroutines)
- Diukur dalam MFLOPs

  Popular for:
  - 100x100 system of equations
  - 1000x1000 system of equations

  Represent:
  - finite element analysis, simulation
  - calls for high computation speed
  - graphics processing

## Dhrystone Benchmark

- Synthetic computing benchmark yang dikembangkan oleh Reinhold P. Weicker pada Siemens [1984] dengan tujuan merepresentasikan pemrograman integer
- Kernel memiliki banyak procedur dan pemanggilan (call procedure) yang percabangannya rendah (low dynamic nesting) dan jumlah instruction per function call yang rendah tetapi waktu komputasi banyak digunakan untuk meng-copy character-string dan membandingkannya
- Weicker mengumpulkan meta-data dari broad range of software, kemudian program-program itu dikarakteristikan dalam bentuk umum: procedure call, pointer indirection, assignment, etc.
- Dalam bahasa C, Ada and Pascal
- Hasil dinyatakan dengan DIPS (Dhrystone Instructions Per Second)
- Benchmark ini digunakan untuk mengukur integer performance



## Lawrence Livermore Loops

- Workload terdiri dari 24 pengujian terpisah yang didominasi oleh perhitungan vektor.
- Diawali di Lawrence Livermore National Laboratories, kemudian dijalankan pada beberapa sistem dari superkomputer sampai PC
- Benchmark yang dihasilkan tidak pernah nilai tunggal karena itu nilai yang diambil berupa maximum, minimum dan tiga mean (Arithmetic, geometric dan harmonic)
- Hasilnya dinyatakan dalam MFLOPS (Million of Floating Point Operation per second)
- Contoh pada :
  - Large Scale science application: floating point calculation in single and double precision arithmetic
  - Large Scale computational fluid dynamic, astrophysic
  - Monte Carlo Simulation



## Debit-Credit Benchmark

- Berupa benchmark berlevel aplikasi sebagai standar Transaction Prrocessing System sejak 1973 sering diserbut TPC Benchmark

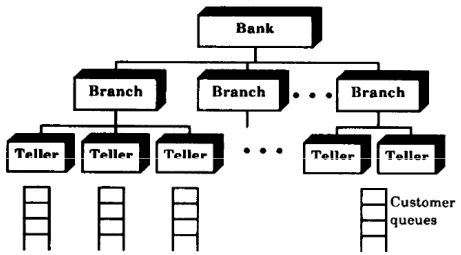


**FIGURE 4.4** Banking environment.

```
Begin-Transaction
    Read message from the terminal (100 bytes)
    Rewrite account                (100 bytes, random)
    Write history                  (50 bytes, sequential)
    Rewrite teller                 (100 bytes, random)
    Rewrite branch                 (100 bytes, random)
    Write message to the terminal  (200 bytes)
Commit-Transaction
```

**FIGURE 4.5** Debit-credit transaction pseudo-code.



## SPEC Suite 1992, 1995, 2000...
## (System Performance Evaluation Cooperative)

1. a nonprofit corporation formed by leading computer vendors to develop a standardized set of benchmarks.
2. Release 1.0 of the SPEC benchmark suite (SPEC 1990) consist of 10 benchmark submitted by scientists and engineers.
   1. *GCC*: The time for the GNU C Compiler to convert 19 preprocessed source files into assembly language output is measured. This benchmark is representative of a software engineering environment and measures the compiling efficiency of a system.
   2. *Espresso*: Espresso is an Electronic Design Automation (EDA) tool that performs heuristic boolean function minimization for Programmable Logic Arrays (PLAs). The elapsed time to run a set of seven input models is measured.
   3. *Spice 2g6*: Spice, another representative of the EDA environment, is a widely used analog circuit simulation tool. The time to simulate a bipolar circuit is measured.
   4. *Doduc*: This is a synthetic benchmark that performs a Monte Carlo simulation of certain aspects of a nuclear reactor. Because of its iterative structure and abundance of short branches and compact loops, it tests the cache memory effectiveness.
   5. *NASA7*: This is a collection of seven floating-point intensive kernels performing matrix operations on double-precision data.

6. *LI*: The elapsed time to solve the popular 9-queens problem by the LISP interpreter is measured.
7. *Eqntom*: This benchmark translates a logical representation of a boolean equation to a truth table.
8. *Matrix300*: This performs various matrix operations using several LINPACK routines on matrices of size 300 × 300. The code uses double-precision floating-point arithmetic and is highly vectorizable.
9. *Fpppp*: This is a quantum chemistry benchmark that performs two electron integral derivatives using double-precision floating-point FORTRAN. It is difficult to vectorize.
10. *Tomcatv*: This is a vectorized mesh generation program using double-precision floating-point FORTRAN. Since it is highly vectorizable, substantial speedups have been observed on several shared-memory multiprocessor systems.

- .



- These benchmarks, which stress primarily the CPU, Floating Point Unit (FPU), and to some extent the memory subsystem, are meant for comparing CPU speeds. Benchmarks to compare I/O and other subsystems may be included in future releases.
- The elapsed time to run two copies of a benchmark on each of the *N* processors of a system (a total of 2N copies) is measured and compared with the time to run two copies of the benchmark on a reference system (which is VAX-11/780 for Release 1.0). For each benchmark, the ratio of the time on the reference system and the system under test is reported as **SPECthruput** using a notation of #CPU@Ratio.
- The TPC and SPEC are the beginning of a new trend in the industry to develop standard benchmarks for comparing all types of computer systems including networks, image processing systems, and databases



## Application Benchmarks

- Specific applications for:
  - Banking
  - Airline reservation
  - large scientific codes

    Includes everything
    - hardware
    - input/output
    - networks
    - operating system
    - databases
- e.g. debit credit benchmark