Lecture 4:

BENCHMARKING:

Benchmarking (in computing) is the act of running a set of computer programs, or other operations in order to assess the relative performance of an object in the computer system. -vedio

A computer must be executing some sort of
program when you attempt to measure any aspect of its performance. Since you are ultimately interested in how the computer performs on your application programs, the best program to run is, obviously, one of your own applications. Unfortunately, this is not always possible since a substantial amount of time and effort may be required to port your existing application to a new computer system. It will perhaps not be cost-effective to port the application if
the only goal is to measure the performance of the new system. Or, it may be that you are evaluating computer systems to determine which one is most appropriate for developing a completely new application. Since the application does not yet exist, it would be impossible to use it as your test program. Owing to these practical and logistical difficulties in running your application program on the system or systems being evaluated, you instead are often forced to rely on making measurements while the computer system is executing some other program. This surrogate program is referred to as a
benchmark program.
since it is used as a standard reference for comparing performance results. The hope is that this standardized benchmark program is in some way characteristic of the applications that you plan to execute on the machine you are

evaluating. If it is, you can use the measurements obtained when executing the benchmark program to predict how well the system will execute your application. The accuracy of thesepredictions determines the quality of a benchmark program, or a set of such programs, for your specific needs. -book 1 pg 190

Benchmark: elaborately-designed benchmarking programs.
• Usually associated with assessing performance of hardware.-vedio
For eg: the floating point operation performance of a CPU.
• But the technique is also applicable to software.-vedio
For eg: the software benchmarks run against compiler or database management system.

missing.....

Purpose of Benchmarking

• Provides a method of comparing the performance of various systems (or subsystems) across different architectures.-vedio

missing...

Benchmarks are designed to mimic a particular type of workload on a component or system.-vedio
• It also helps to choose effective system upgrades and to identify potential bottlenecks.-vedio

missing...

Types of Benchmarks

Since different application domains have different execution characteristics, a wide range of benchmark programs has been developed in the attempt to characterize these different domains. Furthermore, different types of benchmarks satisfy the needs of many different types of users.-book 1 pg 192

i) Addition Instruction

Historically, when computer systems were first introduced, processors were the most expensive and most used
components of the system. The performance of the computer system was synonymous with that of the
processor. Initially, the computers had very few instructions. The most frequent is the addition instruction. -book 5 pg 60

Since almost all of a computer's instructions
required the same amount of time to execute, knowing the time
required to execute a single/addition instruction was sufficient to completely characterize the performance of the system. Quite simply, the machine with the fastest addition operation would produce the best overall performance when executing any application program. -book 1 pg 193 last line

- Historically, the performance of computer system was synonymous with that of the processor.
- Addition was the most frequent instruction.
- The machine with the fastest addition operation would produce the best overall performance when executing any application program. -vedio

ii) Instruction Mixes

To improve system performance, computer architects began to

design processor-instruction sets in which each instruction would
take only the minimum number of cycles required to complete its
particular operation. For example, an instruction that accessed main memory
might take longer than a simple arithmetic operation that accessed only the
registers within the processor. Similarly, an addition instruction would be
executed in less time than a
multiplication or a division instruction. These performanceimprovement
techniques caused processors and systems to become increasingly more
complex. As a result, the execution time of a single instruction was no longer
adequate to summarize performance.-book 1 pg 193

In response to this problem, in the late 1950s Jack C. Gibson
proposed the *Gibson instruction mix* as a benchmarking/performance metric.
The basic idea of this instruction mix is to categorize all of the
instructions into different classes such that each instruction in the
same class requires the same number of processor cycles to
execute. The number of instructions of each class executed by a
particular collection of programs is used to form a weighted average.
This weighted average then is the performance metric used to
compare systems. So Gibson proposed some specific weights for a set of
predefined instruction classes based on measurements of programs running
on IBM 704 and 650 systems.-book 1 pg 194

Gibson instruction mix chart.-book 5 pg 62

Example : Gibson instruction mix.-chart in vedio
so here we can conclude this topic by discussing the
disadvantages of instruction mixes as a benchmark.
1) The emerging processor continued to increase the number of
instructions in there instruction set. Therefore it was very
difficult to include all the instructions in the mix.
2) & the 2nd reason was the introduction of advance features
such as pipelining and address translation mechanism etc. That
was never incorperated in this type of benchmarking.

• Instruction sets in which each instruction take only the minimum number of cycles required to complete its particular operation were designed.
• Execution time of a single instruction was no longer adequate to summarize performance.
• Gibson proposed instruction mix as benchmarking.
• The basic idea was to categorize all of the instructions into different classes.
• Number of instructions of each class executed by the particular collection of programs was then used to form a weighted average. -vedio

iii) Kernel

The introduction of pipelining, instruction caching, and various address translation mechanisms made
computer instruction times highly variable. An individual instruction could no longer be considered in
isolation. Instead, it became more appropriate to consider a set of instructions, which constitutes a higher level
function, a *service* provided by the processors. Researchers started making a list of such functions and using
the most frequent function as the workload. Such a function is called a **kernel**.-book 5 pg 63
A kernel benchmark program is
a small program that has been extracted from a larger application
program.-book 1 pg 197

These benchmark codes are basically designed to measure basic architectural features of parallel machine & they are normally extracted from the actual program.
For instant
It may consist of the inner portion of a loop that consumes
a large fraction of the total execution time of a complete application program,

for instance. It is hoped that, since this loop is executed frequently, it is somehow characteristic of the most important operations performed by the overall application program.
Since a program kernel is typically small, consisting of perhaps only a dozen lines of code, it should be easy to port to many different computer systems. Measuring the performance of this small kernel on several different systems, then, could provide some relative indication of how the different systems would perform when executing the complete application.-book 1 pg 197

Most kernels are processing kernels i.e. only focus is on processor's performance, so these kernels do not issue I/O instructions and therefore measure the performance of the processor without regards to the other system's component.
As a result, kernel benchmarks are of only limited usefulness in making overall comparisons or predictions regarding performance.-book 1 pg 198 last line

• Small benchmark codes designed to measure basic architectural features of parallel machines
• Normally abstracted from actual program
• Easy to port to many different computer systems.
• Most kernels are processing kernels – the only focus is on processor's performance.-vedio

Popular kernels
• Livermore loops consists of 24 do loops, some of which can be vectorized, and
some of which cannot
• Linpack benchmark (contains basic linear algebra subroutine written in FORTRAN
language)
• Tower of Hanoi, 8-Queen Puzzle Problem, Sorting etc.-vedio

iv) Component Benchmark / Micro-benchmark

Microbenchmarks are small, specially designed programs used to test some specific portion of a system.
For example, a small program written to test only the processor-memory interface, the input/output subsystem, or the floating-point-execution unit,independent of the other components of the system, would be a microbenchmark. Microbenchmarks are typically used to characterize the maximum possible performance that could be achieved if the overall system performance were limited by that single component.
Carefully targeted microbenchmarks can be used to determine whether the performance capabilities of all of the components within a system are balanced, for instance. Or, as another example, they are often used to determine the performance characteristics of a system's components to provide values for important parameters for simulation of the system. Writing this type of benchmark typically requires the programmer to have a deep understanding of the system component to be tested.-book 1 pg 197

• Consists of a relatively small specific piece of code to measure performance of computer’s specific portion/component.
• Example: A small program written to test only the processor-memory interface, the input/output subsystem, or the FP execution unit independent of the other components of the system.
• Determine whether the performance capabilities of all the components within a system are balanced.
• Provide values for important parameters for a simulation of the system.
• Requires programmer to have a deep understanding of system being tested.-vedio

Some more examples:

• The most important component in a particular case depends on the individual's usage patterns.
For eg:
• A gamer seeking the best possible frame rates will probably be better served by a faster GPU than by more memory.
Also
• A casual user seeking more responsive system may benefit by upgrading a slow hard drive to fast solid-state drive.
So
• It is necessary to tailor the suite of benchmarks according to user’s specific needs and then weigh the individual test results accordingly.-vedio

v) Synthetic Benchmarks

Synthetic programs were developed to account for this by exercising a mix
• Presents a mix of computations, system calls and I/O requests.
There are
• Artificial programs that mimic a real program execution environment by using statistical data about real high-level language programs.
The main advantage of synthetic program is that they can be
• Can be developed quickly and do not rely on real data.
These programs were written high lavel language to assure portability.
Some examples of the relevant statistical data are:
The statistical data includes
• Fraction of statements of each type (assignment, conditionals, for-loops, etc.)
and

• Fraction of variables of each type (integer, real, character, etc.) and locality (local or global)
local variable or global variable?
• Fraction of expressions with certain number and type of operators and operands.-vedio

• The Procedure for programming synthetic benchmark: is that
• take statistics of all types of operations from many application programs
then
• get proportion of each operation
& then
• write program based on the proportion above.
However these pgms are far from perfect.
The main drawback is that they are
• They are too small to make statistically relevant memory or disk references.-vedio

Similar issues arise with page faults disk or cache hits & misses & the cpu I/O over lap.-book 5 pg 66

vi) Application Benchmarks (or Real programs)

To improve on the limited capabilities of kernel and synthetic benchmarks, standardized sets of real application programs have been collected into various *application-program benchmark suites*. These applications are complete, real programs that actually produce a useful result, in contrast to kernel and synthetic benchmark programs.-book 1 pg 198
These are the small
• A small subset of real-world programs that are representative of the application domain of interest.-vedio

The
Collections of programs are often selected to emphasize one particular class of applications, such as scientific and engineering applications, ‘typical' engineering workstation applications, applications appropriate for parallel-computer systems,
and so forth.-book 1 pg 198
The common applications could be
• e.g. word processing software, tool software of CAD, user's application software (i.e. MIS).-vedio

These real application programs can more accurately characterize how actual applications are likely to use a system than can the other types of benchmark programs. However, to reduce the time required to run the entire set of programs, they often use artificially small input data sets. This constraint may limit the applications' ability to accurately model the memory behavior and input/output requirements of a user's application programs. However, even with these limitations, these types of benchmark programs are the best to have been developed to date.-book 1 pg 198

vii) Database benchmarks
These benchmark are used to
• To measure the throughput and response times of database management systems (DBMS).-vedio

viii) Parallel benchmarks
The parallel benchmarks are used on
• Used on machines with multiple cores, processors or systems consisting of multiple machines.-vedio

Application vs Synthetic benchmarks

Main advantage of synthetic benchmark over application one: is that

• the overall application domain characteristics can be closely matched by a single program.
• while, Application benchmarks give much better measure of real-world performance on a given system,
• the Synthetic benchmarks are useful for testing individual components, like a hard disk or networking device.-vedio

Since a synthetic benchmark not designed to do anything meaningful, may show following strange characteristics:
1) Since the expressions controlling conditional statements are chosen randomly,
• a significant amount of unreachable (or dead) code may be created, much of which can be eliminated by a smart compiler.
in this way
• Code-size becomes highly dependent on quality of compiler.
again
2) Random selection of statements may result in unusual locality properties, defeating well-designed paging algorithms.
& the 3rd pb is that
3) The benchmark may be small enough to fit entirely in the cache and result in an unusually good performance.-vedio

Some Popular Benchmarks

i) Whetstone synthetic benchmark:
• It is Based upon the characteristics of FORTRAN programs doing extensive floating-point computation.
• A single Whetstone instruction is defined as a mix of basic high-level operations, and the results are reported as mega-Whetstones per second.
• This benchmark, although still very popular represents

outdated and arbitrary instruction mixes and thus
• not very useful for machines with a high degree of internal parallelism (pipelining, vectored computation etc.),
• particularly in conjunction with optimizing and parallelizing compilers.-vedio

ii) Dhrystone synthetic benchmark:
• It is written in C,
• & it is designed to represent applications involving primarily integer arithmetic and string manipulation in a block-structured language.
• This benchmark is only 100 statements long, and thus would fit in the instruction cache of most systems.
• Also, since calls only two simple procedures, strcpy() and strcmp(), a compiler that replaces them by inline code can cause a very significant speed-up.
• The results are quoted as Dhrystones/sec, i.e. the rate at which the individual benchmark statements are executed.-vedio

iii) Linpack application benchmark:

• It can Solves a dense 100 x 100 linear system of equations using the Linpack library package.
• Two problems with this benchmark:
• Over 80% of the time is spent in doing A(I) =B(I) +C x D(I) type of calculation, making results highly dependent on how well such a computation is handled,
another pb is that this specific pb is
• Problem is too small.-vedio

Precautions in Benchmarking

A number of hardware and software factors need to be considered before running benchmarks.
1) Many benchmarks place significant stress on specific components,
so we need to ensure
• ensure all such components are in good working order,
• properly cooled (if necessary), and receiving adequate power.
2) When comparing machines using benchmarks, some care is needed to eliminate the effect of extraneous factors.
• for e.g., a highly optimizing compiler can often speedup the code twicely compared with an ordinary compiler.
• Therefore, it’s necessary that benchmarks for all machines be compiled using compilers with similar features and settings.
3) The details of machine configuration should be spelled out clearly, e.g., one could not claim much shorter times by just increasing the cache-size.-vedio
4) Parameters for OS, applications and drivers must be satisfied to ensure accurate, repeatable benchmark results.
• Windows (and other) OSs proactively prefetch data and store numerous temporary files that could interfere with a benchmark, so it's best to clear out any temporary files and prefetch data before running a test.
• In Windows 7, you can find prefetch data in
C:\Windows\Prefetch, and temporary files in
C:\Windows\Temp and
C:\Users\[username]\AppData\Local\Temp.
So the prefetching can actually optimize the final results & can generate the misleading outcome.
5) Using the correct (or latest) drivers for a component to ensure it is operating and performing optimally.

• especially true of graphics boards and motherboards /chipsets, where wrong driver can significantly worsen the system's frame rates or transfer speeds and latency.
• Finally, confirm that the OS is fully updated and patched to ensure optimal compatibility and to reflect the current, real-world OS configuration.-vedio

List of Challenges

Interpretation of benchmarking data is extraordinarily difficult.
1) Manufacturers commonly report only those benchmarks (or aspects of benchmarks) that show their products in the best light.
2) Many benchmarks focus entirely on the speed of computational performance, neglecting other important features such as Qualities of Service.
• Transaction Processing Performance Council (TPPC) Benchmark specifications partially address these concerns by specifying ACID property tests, database scalability rules and service level requirements.-vedio
Actually this is very important to consider that just the speed of operation is not sufficient we need to cater the other requirements as well so the TPPC benchmark specification is actually highlighting this point by specifying that the vendors should give the ACID property tests results, the scalability rules results and service level requirements as well to the customer.
3) Users can have very different perceptions of performance than benchmarks may suggest.
for example
• In particular, users appreciate predictability — servers that always meet or exceed service level agreements.
therfore

• Benchmarks tend to emphasize mean scores (IT perspective), rather than maximum worst-case response times (real-time computing perspective), or low standard deviations (user perspective).
4) Many server architectures degrade dramatically at high (near 100%) levels of usage.
• Vendors, in particular, tend to publish server benchmarks at continuous at about 80% usage — an unrealistic situation.-vedio

Benchmark Strategies

Most of the types of benchmark programs base the measure of performance on time required to execute benchmark. There are Several other strategies can be employed in a benchmark program, however.
Specifically, the three different strategies for using a benchmark program are the following.-vedio

1) Fixed Computation Benchmarks:
This type of benchmark measures the time required to execute a fixed workload.
Measure the time required to perform a fixed amount of computation.
• a fixed workload
• Example: The SPEC (Standard Performance Evaluation Corporation) CPU benchmarks
• www.spec.org see different egs in this site.-vedio

2) Fixed Time Benchmarks:

An alternative view argues that users with large problems to solve

are willing to wait a fixed amount of time to obtain a solution..-book 1 pg 204
Thus, instead of holding the amount of computation performed by the benchmark constant, the amount of time that the benchmark program is allowed to run is held constant.book 1 pg 204
At the end of the allotted execution time, the total amount of computation completed, which must be carefully defined, is used as the measure of the relative speeds of the different systems.-book 1 pg 205

Measure the amount of computation performed within a fixed period of time.
• Argument: users with large problems are willing to wait a fixed amount of time to obtain a solution.
• At the end of allotted execution time, total amount of computation completed, used as the measure of relative speeds of different systems.
• Example: The TPC (Transaction Processing perf. Council) benchmarks.
• www.tpc.org -vedio

3) Variable-Computation & Variable-Time Benchmarks:

The third and most general benchmarkprogram strategy fixes neither the available time nor the amount of computation to be done. These types of benchmarks instead try to measure some other aspect of performance that is a function of the computation and the execution time. This derived metric is then
used as the measure of performance when executing the benchmark.-book 1 pg 208
The HINT benchmark is a good example of this variablecomputation, variable-time benchmark-program strategy. It rigorously defines the 'quality' of a solution for a given mathematical problem that is to be solved by the benchmark program. The solution's quality, based on the problem's definition, can be continually improved by doing additional computation. The ratio of this quality metric to the time required to achieve that level of quality

is then used as the measure of performance. The HINT benchmark expresses this final performance metric as QUIPS, or *quality improvements per second*.-book 1 pg 208

Allow both the amount of computation performed and the time to vary.
• These types of benchmarks instead try to measure some other aspect of performance that is a function of the computation and the execution time.
• Example: The HINT (Hierarchical INTegration) benchmark.
• HINT uses a performance measure called QUIPS (QUality Improvements Per Second).-vedio
The system being evaluated is .... to continually improve the quality of solution for specific mathematical problem.