Performance Metrics and Scalability Analysis

Performance Metrics and Scalability Analysis

Lecture Outline

Following Topics will be discussed

- Requirements in performance and cost
- Performance metrics
- Work load and speed metrics
- Communication overhead measurement Analysis
- Theoretical concepts on Performance and Scalability
  - Phase Parallel Model
  - Speed up, Efficiency, Isoefficiency models
  - Examples

Performance Versus Cost

Questions to be answered

- What are the user's requirement in performance and cost?
- How one should measure the performance of application programs?
- What kind of performance metrics should be used?
- What are the factors (parameters) affecting the performance?
- How can one determine the scalability of a parallel computer executing a given application?

Performance Versus Cost

(Contd..)

How do we measure the performance of a computer system?

- Many people believe that execution time is the only reliable metric to measure computer performance

Approach

- Run the user's application time and measure wall clock time elapsed

Remarks

- This approach is some times difficult to apply and it could permit misleading interpretations.
- Pitfalls of using execution time as performance metric.
  - Execution time alone does not give the user much clue to a true performance of the parallel machine

Performance Requirements

Types of performance requirement

Six types of performance requirements are posed by users:

- Executive time and throughput
- Processing speed
- System throughput
- Utilization
- Cost effectiveness
- Performance / Cost ratio

Remarks : These requirements could lead to quite different conclusions for the same application on the same computer platform

Performance Requirements

(Contd..)

Performance versus cost

- Execution time is critical to some applications
  - A real time application, the user cares about whether the job is guaranteed to finish within a time limit
  - **Example :** Radar signal processing application

Processing Speed

- For many applications, the users may be interested in achieving a certain processing speed rather than an execution time limit
  - **Example :** Radar signal processing application

Performance Requirements

(Contd..)

System Throughput

- Throughput is defined to be the number of jobs processed in a unit time

  Example : STAP Benchmarks, TPC Benchmarks

- Remark : Execution time is not only performance requirement, but other performance requirement is needed.

Performance Requirements

(Contd..)

Utilization and Cost Effectiveness

- Instead of searching for the shortest execution time, the user may want to run his application more cost effectively
- High percentage of the CPU hours to be used for useful computing
- Reduction of time spent in load imbalance and communication overheads

Performance Requirements

(Contd..)

Utilization and Cost Effectiveness

- A good indicator of cost-effectiveness is the utilization factor which is ratio of the achieved speed to the peak speed of a given computer.

Performance/Cost Ratio

- Performance/cost ratio of a computer system is defined as the ratio of the speed to the purchasing price.

Performance Requirements

(Contd..)

Remarks

- Higher Utilization corresponds to higher Gflop/s per dollar, provided if CPU-hours are changed at a fixed rate.
- A low utilization always indicates a poor program or compiler.
- Good program could have a long execution time due to a large workload, or a low speed due to a slow machine.
- Utilization factor varies from 5% to 38%. Generally the utilization drops as more nodes are used.
- Utilization values generated from the vendor's benchmark programs are often highly optimized.

Performance Versus Cost

The misleading peak performance / cost ratio

- Using the peak performance/cost ratio to compare system is often misleading

  Example : The peak performance cost ratio of some supercomputers is much lower than those of others. But its sustained performance is actually higher.

- Often, users need to use more than one metric in comparing different parallel computing system
  - The cost-effectiveness measure should not be confused with the performance/cost ratio of a computer system
  - If we use the cost-effectiveness or performance / cost ratio alone, the current Pentium PC easily beat more powerful systems

Performance Versus Cost

(Contd..)

Summary

- The execution time is just one of several performance requirements a user may want to impose.
- The other include *speed, throughput, utilization, cost-effectiveness, performance/cost ratio, and scalability.*
- Usually, a set or requirements may be imposed.

Workload and Speed Metrics

Workload and speed metrics

- Three metrics are frequently used to measure the computational workload of a program
  - **The execution time** : The first metric is tied to a specific computer system. It may change when the C program is executed on a different machine.
  - **The number of instructions executed** : The second metric, tied to an instruction set architecture (ISA), stays unchanged when the program is executed on different machines with same ISA.
  - **The number of floating-point operations executed** : The third metric is often architecture-independent.

Workload and Speed Metrics

(Contd..)

Workload and Speed Metrics

Summary of all three performance metrics

Workload Type	Workload Unit	Speed Unit
Execution time	Seconds (s), CPU clocks	Application per second
Instruction count	Million instructions or billion instruction	MIPS or BIPS
Floating-point operation (flop) count	Flop, Million flop (Mflop), billion flop (Gflop)	Mflop/s Gflop/s

Workload and Speed Metrics

(Contd..)

Instruction count

- The work load is the instructions that the machines executed, not just the number of instructions in assembly program text.
- Instruction count may depend on the input data values. For such an input dependent program, the workload is defined as the instruction count for the **worst-case** input.

Example

A sorting program may execute 100000 instructions for a given input, but only 4000 for another. For such an input-dependent program, the workload is defined as the instruction count for the worst case input.

Workload and Speed Metrics

(Contd..)

Instruction count

- Even with a fixed input, the instruction executed could be different on different machines.
- Even with a fixed input, the instruction count of a program could be different on the same machine when different compilers or optimizations are used.
- Finally, a larger instruction count does not necessarily mean the program needs more time to execute.

Workload and Speed Metrics

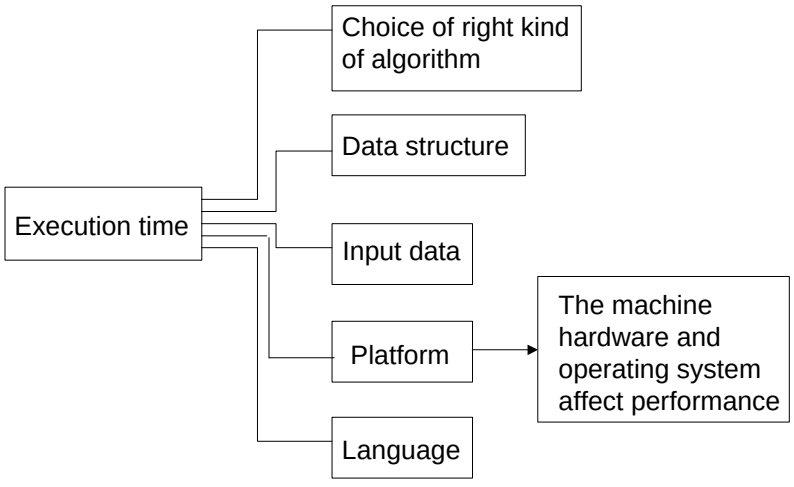
(Contd..)

Execution time

- For a given program on a specific computer system, one can define the workload as the total time taken to execute the program.This execution time should be measured in terms of wall clock time, also known as the elapsed time.
- Execution time depends on many factors explained below.
- The basic unit of workload is seconds.

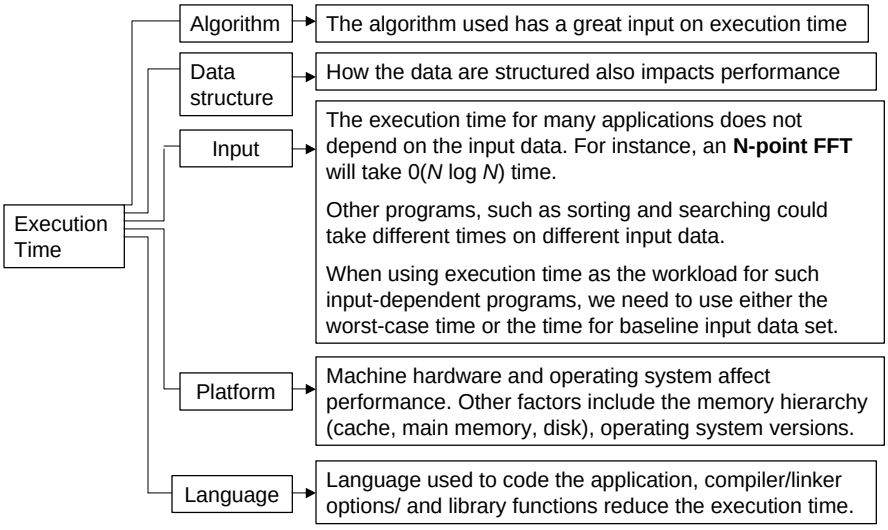
Workload and Speed Metrics

(Contd..)



Workload and Speed Metrics

(Contd..)



Workload and Speed Metrics

(Contd..)

Floating point count

- When application program is simple and its workload is not input-dependent, the workload can be determined by code inspection.
- When the application code is complex, or when the workload varies with different input data (e.g., sorting) one can measure the workload by running the application on a specific machine.
- This specific run will generate a report of the number of flops or instructions actually executed.
- This approach has been used in the NAS benchmark, where the flop count is determined by an execution run time.

Workload and Speed Metrics

(Contd..)

Rules for Counting Floating-Point Operations (NAS standards)

Operations	Flop Count	Comments on Rules
A[2*1] = B[j -1]+1.5*C - 2;	3	Add, subtract, or multiply each count as 1 flop Index arithmetic not counted Assignment not separately counted
X = Y;	1	An isolated assignment is counted as 1 flop
If (X>Y) Max = 2.0*X;	2	A comparison is counted as 1 flop
X = (float) i + 3.0;	2	A type conversion is counted as 1 flop
X = Y / 3.0 + sqrt(Z)	9	A division or square root is counted as 4 flop
X = sin(Y) - exp(Z);	17	A sine, exponential, etc. is counted as 8 flop

Workload and Speed Metrics

(Contd..)

Caveats

- Using execution time or instruction count as the workload metric leads to a strange phenomenon.
- The workload changes when the same application is run on different systems.
- Workload can be determined only by executing the program.
- The flop count and the instruction count have another advantage: their corresponding speed and utilization measures give some clue as to how well the application is implemented.
- Highly tuned programs could achieve more. It is possible to achieve 90% utilization in some numerical subroutines.

Workload and Speed Metrics

(Contd..)

Summary of performance metrics

- To sum up, all three metrics are useful, but especially the flop count and the execution time.
- In practice, the flop count workload is often determined by code inspection, as described in the table.
- The execution time is often measured on a specific machine, under a set of specific listing conditions including hardware platform, compiler options, and input data set etc.

Remark : The flop count metric is more stable.

Computational Characteristics of Parallel Computers

Computational Characteristics

The historical values of performance parameters for PVP, SMP, MPPs and Clusters are

- Year of release in Market
- Clock Rate (MHz)
- Memory Capacity
- Machine Size
- Peak performance per node (Mflop/sec)
- Peak performance on *n* (> 1) nodes (Gflop/sec)

Computational Characteristics of Parallel Computers

(Contd..)

The Memory Hierarchy

For each layer, the following three parameters play a vital role for extracting performance

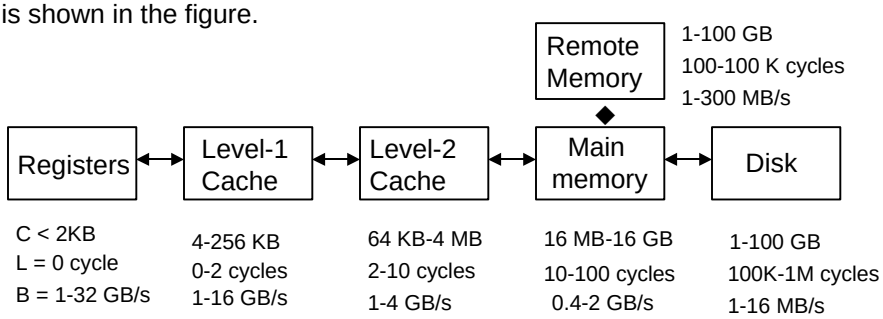
- Capacity (C) : How many bytes of data can be held by the device ?
- Latency (L) : How much time is needed to fetch one from the device ?
- Bandwidth (B) : For moving a large amount of data, how many bytes can be transferred from the device in a second ?

Computational Characteristics of Parallel Computers

(Contd..)

The Memory Hierarchy

The performance depends on how fast the system can move data between processors and memories. The memory subsystem hierarchy is shown in the figure.



Performance parameters of a typical memory hierarchy

Computational Characteristics of Parallel Computers

(Contd..)

- The faster and smaller devices are closer to the processor.
- The devices closest to the processor are the registers, which are in fact part of the processor chip.
- The level - 1 cache is usually on the processor chip and the level – 2 cache is off chip.
- The level – 3 cache is off chip shared by some processors in SMP node.
- The main memory includes the local memory in a node, and the global memory for the machines with a centralized shared memory.

Parallelism and Interaction Overheads

The time to execute a parallel program is

$$T = T_{comp} + T_{par} + T_{interact}$$

where T_{comp}, T_{par} and $T_{interact}$ are the times needed to execute the computation, the parallelism, and the interaction operations, respectively. (*Assume that interaction overheads are ignored, and no overlapping is involved*).

Parallelism and Interaction Overheads

(Contd..)

Source of Parallelism overhead

- Process management, such as creation, termination, context switching, etc.
- Grouping operations, such as creation or destruction of a process group
- Process inquiry operations, such as asking for process identification, rank, group, identification, group size, etc.

Parallelism Overheads

- Creating a process or a group is expensive on parallel computers. This is important for the parallel program which is executed many times to process raw data.

Parallelism and Interaction Overheads

(Contd…)

Sources of interaction overheads

- Synchronization, such as barrier, locks critical regions, and events
- Aggregation, such as reduction and scan
- Communication, such as point-to-point and collective communication and reading/writing of shared variables
- Idleness due to Interaction

Parallelism and Interaction Overheads

(Contd..)

Overheads Quantification

- Measurement conditions
- Measurement methods
- Expressions for overhead parallelism due to communications
- Point-to-Point Communication Communication overhead for message length *m* (in bytes) (*requirement of startup time, asymptotic bandwidth*)
- Collective communication and Collective computation

Parallelism and Interaction Overheads

(Contd..)

Overhead measurement conditions

The following conditions are used to estimate the overhead involved.

- The data structures used (The data structures used are always made small enough to fit into the node memory so that there will not be page faults)
- The programming language used.
- The best complier options should be used.
- The message passing library used.
- The communication hardware and protocol used.
- To use wall clock time elapsed.

Parallelism and Interaction Overheads

(Contd..)

Overhead measurement Methods

- Measuring point-point communication (ping-pong test) between two nodes
- Measuring point-point communication involving *n* nodes and collective communication performance

Interpretation of overhead measurement data

- Method 1 : Present the results in tabular form
- Method 2 : Present the data as curve
- Method 3 : Present the data using simple, closed-form expression that can be used to estimate the overhead for various message lengths and machine size.

Parallelism and Interaction Overheads

(Contd..)

Remarks

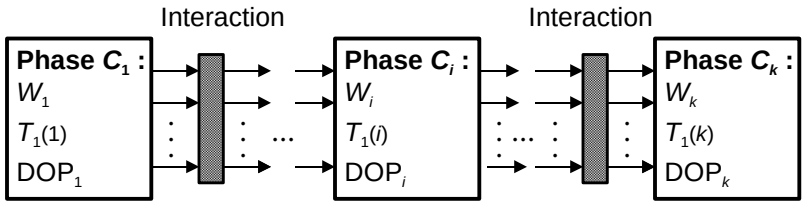
- Knowing the overheads can help programmer decide how to best develop a parallel program on a given parallel computer.
- Parallelism and interaction overheads are often large compared to the basic computation time, and they vary from a system to system.
- In some cases parallelism and interaction overheads are significant and they should be quantified.
- The overhead values also vary greatly from one system to another.

Performance Metrics : Phase Parallel Model

Phase Parallel model : **Analysis**

Consider a sequential program *C* consisting of a sequence of *K* major component computational phases C_1, C_2, C_3... C_K.

We want to develop an efficient parallel program *C* by exploiting each phase C_i with a Degree Of Parallelism DOP_i. A phase parallel program is depicted in the following figure



The phase parallel model of an application algorithm

Performance Metrics: Phase Parallel Model

(Contd..)

Phase Parallel model : **Basic Metrics**

While executing on *p* processors with $1 \le p \le DOP_i$, we have

- The parallel execution time for step Ci is Tp(i)=T1(i) / p,
- The total parallel execution time over p nodes becomes

$$T_n = \sum_{1 \le i \le k} \frac{T_1(i)}{\min(DOP_i, n)} + T_{par} + T_{interact}$$

- T_{par} and $T_{interact}$ denote all parallelism and interaction overheads.

Performance Metrics: Phase Parallel Model (Contd..)

Remark

1. In many cases, the total overhead is dominated by the communication overhead.
2. The implementation of point-to-point and collective communication and computation overhead for various message sizes in MPI play a important role for this overhead.
3. These metrics will help the Application User to know about performance issues of a particular code.
4. These inequalities are useful to estimate the parallel execution time

Performance Metrics: Phase Parallel Model (Contd..)

Phase parallel model

Notation	Terminology	Definition
T_1	Sequential time	$T_1 = \sum_{1 \le i \le k} T_1(i)$
T_p	Parallel time, p-node time	$T_p = \sum_{1 \le i \le k} \frac{T_1(i)}{\min(DOP_i, p)} + T_{par} + T_{interact}$
T_∞	Critical path	$T_\infty = \sum_{1 \le i \le k} \frac{T_1(i)}{DOP_i}$
S_p	Speedup	$S_p = T_1 / T_p$
P_p	p-node speed	$P_p = W / T_p$
E_p	p-node efficiency	$E_p = S_p / p = T_1 / (pT_p)$
T_0	Total overhead	$T_0 = T_{par} + T_{interact}$

Performance Metrics of Parallel Systems

- **Speedup :** Speedup T_p is defined as the ratio of the serial runtime of the **best** sequential algorithm for solving a problem to the time taken by the parallel algorithm to solve the same problem on *p* processor
- The *p* processors used by the parallel algorithm are assumed to be **identical** to the one used by the sequential algorithm
- **Cost :** Cost of solving a problem on a parallel system is the product of parallel runtime and the number of processors used

$$E = p.S_p$$

Performance Metrics of Parallel Systems (Contd..)

- **Efficiency :** Ratio of speedup to the number of processors.
- Efficiency can also be expressed as the ratio of the execution time of the **fastest** known sequential algorithm for solving a problem to the **cost** of solving the same problem on *p* processors
- The **cost** of solving a problem on a single processor is the execution time of the known best sequential algorithm
- **Cost Optimal :** A parallel system is said to be cost-optimal if the cost of solving a problem on parallel computer is proportional to the execution time of the fastest known sequential algorithm on a single processor.

Performance Metrics of Parallel Systems

(Contd..)

- Cost optimal parallel system has an **efficiency** of θ (1)
- Cost is sometimes referred to as work or processor-time product and a cost-optimal systems is also known as PT_P optimal system
- Using fewer than the maximum number of processors to execute a parallel algorithm is called **scaling down** a parallel system in terms of number of processors

Scalability and Speedup Analysis

(Contd..)

A Ideal Scalability metric should have the following two properties

1. It predicts the workload growth rate with respect to the increase of machine size.
2. It is consistent with execution time, i.e., a more scalable system always has a shorter execution time.

Remark :

- It can be shown that under very reasonable conditions, a system with a small **isoutlization** (thus more scalable) always has a shorter execution time.
- Systems with a small **isoutilization** are more scalable than those with a large one.

Conclusions

Summary of Performance metrics

- Performance metrics and measurement of performance of a parallel program have been explained.
- Work load and speed metrics are explained
- Types of overhead in parallel computing are discussed.
- Theoretical concepts on Performance and Scalability
  - Phase Parallel model, Speed up, Efficiency, Isoefficiency, Isospeed and Isoutilization metrics are explained.

References

1. Ernst L. Leiss, Parallel and Vector Computing A practical Introduction, McGraw-Hill Series on Computer Engineering, Newyork (1995).
2. Albert Y.H. Zomaya, Parallel and distributed Computing Handbook, McGraw-Hill Series on Computing Engineering, Newyork (1996).
3. Vipin Kumar, Ananth Grama, Anshul Gupta, George Karypis, Introduction to Parallel Computing, Design and Analysis of Algorithms, Redwood City, CA, Benjmann/Cummings (1994).
4. William Gropp, Rusty Lusk, Tuning MPI Applications for Peak Performance, Pittsburgh (1996)
5. Ian T. Foster, Designing and Building Parallel Programs, Concepts and tools for Parallel Software Engineering, Addison-Wesley Publishing Company (1995).
6. Kai Hwang, Zhiwei Xu, Scalable Parallel Computing (Technology Architecture Programming) McGraw Hill Newyork (1997)
7. Culler David E, Jaswinder Pal Singh with Anoop Gupta, Parallel Computer Architecture, A Hardware/Software Approach, Morgan Kaufmann Publishers, Inc, (1999)

Thank you