Proceedings of the 5th WSEAS Int. Conf. on Software Engineering, Parallel and Distributed Systems, Madrid, Spain, February 15-17, 2006 (pp211-216)

A Queueing Model for Evaluating the Performance of Parallel Processing Systems

CHANINTORN JITTAWIRIYANUKOON
Department of Telecommunications Science
Faculty of Science and Technology
Assumption University

Abstract:-This paper presents a queueing model to measure the performance of parallel processing network by introducing the 80 parallel computers for all subtasks execution. First, the case in which a task with granularity is discussed. The parallel system improves the performance by distributing and executing subtasks on dedicated 80 parallel computers. Delays in task partitioning, subtask distribution, communications and merging are taken into account for this performance evaluation. However, the prototype of this parallel environment is indeed costly. In this paper a cost-effective queueing model is proposed in order to investigate how parallel systems can function, as tasks with granularity exist. Results from cost-effective model are firstly crosschecked to results collected from 80 parallel computing units. By altering task characteristics for the mentioned parallel computing system, we can obtain the supportive results from the cost-effective queueing model.

Keywords:-Parallel processing, granularity, parallelism, queueing model, performance evaluation.

1.Introduction

Computer networking evolved from telecommunications terminal-computer communication, where the object was to connect remote terminals to a central computing facility. Since computer networks were invented, the number of applications that can be implemented on the computer networks have been changing steadily from sending a simple electronic mail, to complicated areas such as parallel and distributed computing [2]. The computer networks are becoming an integral part of every society. Ongoing researches will continue to improve the data transfer rate to make it even faster and more reliable. The application and software designers cannot emphasize only on the data exchange, they must find a better way to increase the current computing power. Research on parallel and distributed system on computer network may provide the answer.

For the last decade, the computers evolution has been changed drastically both software and hardware. On the hardware side, the development is concentrated on "Multi-processor", a scheme which more than one CPUs are loosely working together to perform an assigned task. This kind of development is expected to continue in a foreseeable future. Although it is still expensive to implement, we still can use this multiprocessor scheme to improve the efficiency of a computer system. Another alternative is to use parallel processing (parallel processing is a process that utilizes a number of processors to compute subtasks simultaneously) to shorten the overall processing time and to improve the efficiency. This paper will emphasize on parallel processing network to improve the processing speed and task distribution by utilizing a client/server architecture.

The main goal of parallel computation is to increase the processing speed. This can be obtained by having multiprocessors to perform each subtasks as a parallel processing units. There are two parallel architectures that are currently used. These are single instruction multiple data stream (SIMD) and multiple instruction multiple data stream (MIMD). However, to design a low cost parallel system is to employ the facilities of computer network. Another method in the past was Parallel Virtual Machine (PVM) project which implemented a concurrent system onto multiplatform workstations [3].

In order to develop a client/server architecture, initially, all clients are to be autonomous systems in the network. A protocol can control all clients to work closely together in parallel fashion. Parallel system built by the client/server network has no problem of communication delay as the bandwidth of current computer network is as high as 1 Gbps. But major problems may arise whenever the partitioning into subtasks is performed. Subtasks then are distributed to

clients. Each client has processing capability per se plus a separate memory unit to process to task. The subtask programs on clients can be identical or different in size. Once subtask processing is completed then result will be forwarded to the server. All subtasks will wait at the server until the last result from client has been completed. All results will then be merged to mark the completion of parallel processing.

2.Client/Server Architecture

A.TCP/IP

TCP/IP (**Transmission Control Protocol/Internet Protocol**) is one of the most popular protocols firstly developed by a community of researchers center at the ARPAnet. TCP/IP is a connectionless protocol. Information will be packetized prior to the transmission. Each of these packets is transmitted through the network individually. TCP is responsible for ensuring the commands can get through to the other end. It keeps track of what is sent, and retransmits if necessary. Generally, TCP/IP function comprises of 4 layers:

- an application layer, a protocol deals with applications such as mail, ftp etc.
- TCP layer, a protocol provides services needed by many applications.
- IP layer, a protocol needed to manage a specific physical medium, such as Ethernet or a point to point line. This layer also provides the basic service of transferring packets to the destination.
- Physical layer, the layer specifies the characteristic of physical medium.

B. Distributed Model

The model this research employs is distributed system model [1] according to client/server architecture. The internodes communication is done by using RPC (Remote Procedure Call) as message passing system. In the model, job scheduling is an independent queue as each job has no relation to each other in order to perform a parallel computation after partitioning process. The parallel computation programming model can be classified by Task-Farming as shown in figure 1. Since this paradigm is consisting of two entities which are Master(server) and Slave(client). The server takes care of decomposing the process into subtasks and distributes them to all clients. After all, server has to collect all subtasks result from each clients then merge them for completed result.

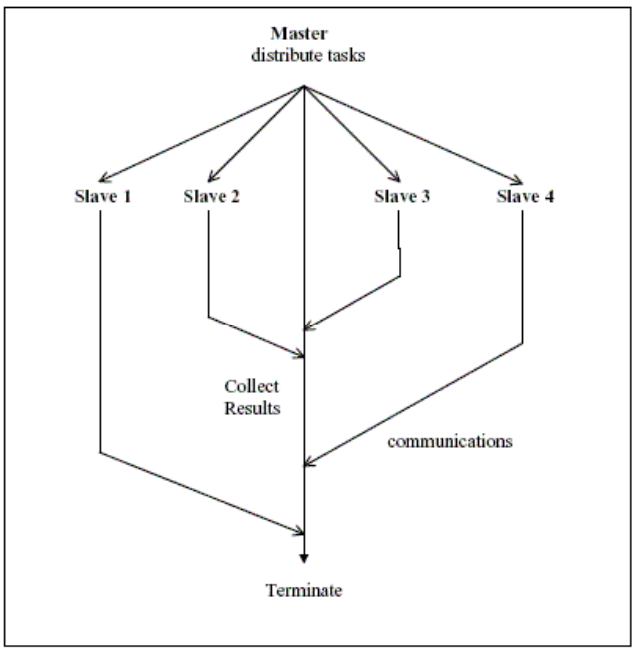

Figure 1. A client/server computation.

The implementation of parallel computing process is to calculate the polynomial function as indicated in table 1. The complexity is classified into 3 categories, low, medium and high load. In order to understand the parallel processing, the system is divided into 2 phases, that is, server and client phases.

- Sever phase: Server firstly uses TCP/IP protocol to poll the entire available clients. Server will then decompose task (polynomial function) based upon the number of available clients. Server will distribute subtasks to entire clients equally. Finally server will collect subtasks results from clients and merge them into the final result of the computation.
- Client phase: Client will be waiting for a handling of subtask from server. After client receives subtask, it will immediately execute the polynomial function. When client finishes execution of designated job, it will communicate with server to transfer the subtask result back to the server. Each subtask result will be tagged by client using client IP address in order that server can recognize the sequence of result to perform merging. Clients then are released and wait for other executions from the server. The client/server model with n client nodes is indicated in figure 2.

Light load	$N(100^{100})$
	$N(100^{200})$
	$N(100^{300})$
Medium load	$N(300^{300})$
	$N(300^{400})$
	$N(100^{500})$
High load	$N(200^{600})$
	$N(200^{800})$
	$N(400^{1000})$

Table 1. Polynomial function for N client nodes.

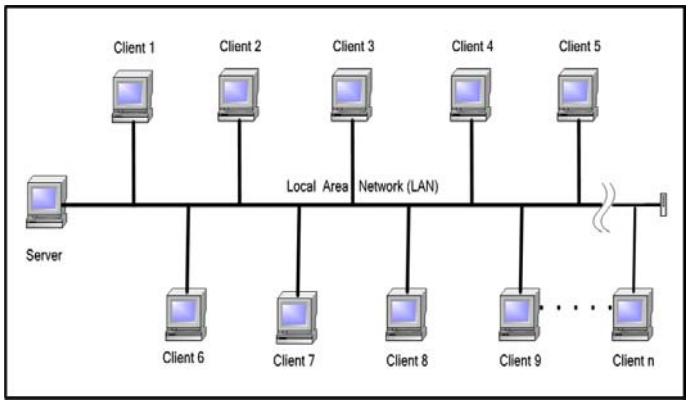

Figure 2. A client/server architecture with n clients.

C. Queueing Model

Queuing theory is the study of the waiting times, lengths, and other properties of queues expressed in mathematical terms [4]. A queue can be simply defined as a waiting line of objects that wait for a service. These objects can be customers at a store waiting for a cashier, a patient waiting at a medical clinic, packets of data in a network, or requests from terminals to a server. In most cases, six basic characteristics of queueing processes provide an adequate description of a queueing system [5]. The six basic characteristics are arrival pattern of customers, service pattern of servers, queue disciplines, system capacity, number of servers, and service classes. Notation in queueing process, which is now the standard, can be described by a series of symbols and slashes such as *A/B/X/Y/Z*, where *A* represents the interarrival time distribution, *B* represents the service pattern as described by the probability distribution for service time, *X* is the number of parallel servers, *Y* is the restriction on system capacity, and *Z* is the queue discipline. Generally the notation can solely be *A/B/X* while *Y* and *Z* are optional. The default value for *Y* and *Z* are applicable to infinity, which means the capacity is unlimited and so is the queue size. Queueing model for this client/server architecture is shown in figure 3. The notation is indicated in table 2.

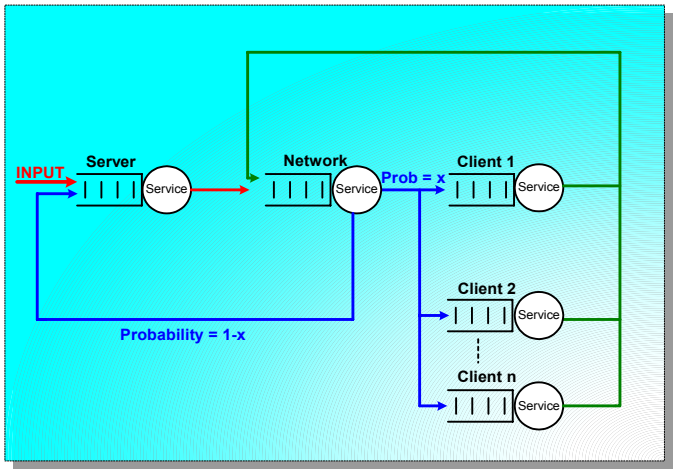

Figure 3. Queueing model.

Characteristics	Symbol	Explanation
Interarrival time distribution (*A*)	M	Exponential Fn
	D	Deterministic
	E_k	Erlang *k* (*k* = 1,2,…)
Service time distribution (*B*)	H_k	Mixture of *k* exponentials
	PH	Phase Type
	G	General
Number of parallel servers (*X*)	1,2, … , ∞	
Restriction on system capacity (*Y*)	1,2, … , ∞	
Queue disciplines (*Z*)	FCFS	First come, first served
	LCFS	Last come, first served
	RSS	Random selection for service
	PR	Priority
	GD	General Discipline

Table 2. Queueing notation *A/B/X/Y/Z.*

3.Input Parameters

All requests arrive in queue prior to server as shown in figure 3. These requests will include requests from clients (in transferring subtasks result) and from server itself (in polling for available clients). Queue capacity in our experiment is assumed to be infinite. The following equation is employed in order to analyze the service time at server (*Ss*).

$$Ss = G + C + D + G + F \quad (1)$$

where *Ss* = Service time at server

G = Polynomial generation time
C = Client nodes polling time
D = Subtask distribution time
G = Subtasks collection time
F = Results merging time

Parameter G (Polynomial generation time): The polynomial function is assumed to reside in the server for parallel processing beforehand then this parameter is set to be 0 in the analysis.

Parameter C (Client nodes polling time): This parameter can be estimated by utilizing the Sniffer program to capture packets in the client/server reference model as shown in figure 2.

Parameter D (Subtask distribution time): By using sniffer program (Sniffer Pro version 4.7) in the client/server reference model to check the average subtask distribution time then the value is normalized per node basis.

Parameter G (Subtasks collection time): This parameter is estimated by using Sniffer program to capture the entire packets produced by clients. The average time of the slowest packets from clients is calculated.

Parameter F (Results merging time): This parameter depends on the processing power of the server in figure 2 and can be estimated by using PAD (Parallel And Distributed) Time calculation program. The "sum" time is taken into the account to represent that server will spend some time for results merging.

The queueing model will refer to 9 cases of polynomial functions as shown in table 1. but number of clients can scale up to 200. An example of input parameters for 9 polynomial functions is listed in table 3.

No. Clients	Ss	G	C	D	R	F
200	0.172446371	0	0.000001	0.1704	2.652E-07	0.002045106
160	0.172957714	0	0.000001	0.1704	3.315E-07	0.002556383
120	0.173809952	0	0.000001	0.1704	0.000000442	0.00340851
80	0.175514428	0	0.000001	0.1704	0.000000663	0.005112765
72	0.176082587	0	0.000001	0.1704	7.36667E-07	0.005680851
64	0.176792786	0	0.000001	0.1704	8.2875E-07	0.006390957
56	0.177705898	0	0.000001	0.1704	9.47143E-07	0.007303951
48	0.178923381	0	0.000001	0.1704	0.000001105	0.008521276
40	0.180627857	0	0.000001	0.1704	0.000001326	0.010225531
32	0.183184571	0	0.000001	0.1704	1.6575E-06	0.012781914
24	0.187445762	0	0.000001	0.1704	0.00000221	0.017042552
16	0.195968142	0	0.000001	0.1704	0.000003315	0.025563827
8	0.221535285	0	0.000001	0.1704	0.00000663	0.051127655

Table 3. Example of service time (sec) at server for polynomial function 100^200.

The request of network service time (*Sn*) arrives from both clients and server. Server and clients in the queueing model (as shown in figure 3.) will share the only communication service. After network service, packets will be routed using probability *k* in order to distribute subtasks to client or using probability *(1-k)* if communication service fails. and server. The network service time arises from packet arrival at the network service, after a suitable management time (overhead), then an acknowledgement must be sent back to the server once packets arrive client nodes. Network service time can be modeled as shown in figure 4. and derived by the following equation.

$$Sn = Mt + Pt + C + L/V + Pr + Mr \qquad (2)$$

where *Sn* = Network service time
Mt = Management time of the transmitter
Mr = Management time of the receiver
Pt = Propagation delay of transmitter
Pr = Propagation delay of receiver
C = Carrier holding time
L = Total length of the packet
V = Capacity of the communication line

The important notice is that the arrivals of the packets come from both server and client. So the packets arrive in a queue with a rate λ and served at a rate μ. The model is represented by figure 4.

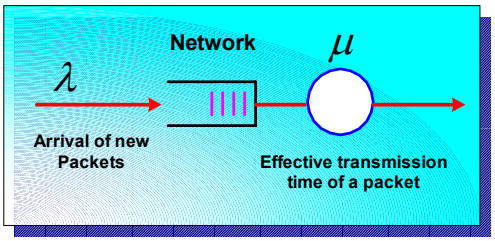

Figure 4. Network queueing model.

Parameter Mt (Management time of the transmitter): Management time on the sender side are necessary for encoding and monitoring, etc. Normally management time on the sender side will be longer than management time at the receiver side.
Parameter Mr (Management time of the receiver): Management time on the receiver side is necessary for decoding and monitoring, etc.
Parameter Pt (Propagation delay of transmitter): Propagation delay is calculated by the standard of IEEE 802.3.
Parameter Pr (Propagation delay of receiver): Propagation delay is also calculated by the standard of IEEE 802.3.
Parameter C (Carrier holding time): is the network equipment delay, for example, network interface card (NIC). This delay is constant and assumed to be 0.00025 second.
Parameter L (Total length of the packet): This parameter came from average number of packet in the reference model (9 polynomial functions) by using sniffer program to capture the entire package during experiment. The length of the packet sent in the network is 663 bytes.
Parameter V (Capacity of the communication line): In figure 2. the capacity of communication line is 100 Mbps. The figure will be used in the queueing model for the analysis in order that both models are exactly identical. The sample of network delay which is estimated from equation 2 is shown in table 4.

Sn	***Mt***	***Mr***
1.66E-2	0.013	0.003
Pt= Pr	***C***	***L/V***
2.56E-6	2.5E-4	4.24E-4

Table 4. Network service time for polynomial function.

Client requests will arise whenever clients try to respond after subtask execution is completed. The client service time (*Sc*) is estimated as written in the equation 3.

$$Sc = A + W + C + S \quad (3)$$

where Sc = Client service time
A = Acknowledge time
W = Waiting time for data from server
C = Computation time
S = Result transmission time

Parameter A (Acknowledge time): the acknowledge time to respond server through communication service.
Parameter W (Waiting time for data from server): this is the time interval client has to wait after polling until data from server arrives.
Parameter C (Computation time): this is time used by a single client to execute subtask.
Parameter S (Result transmission time): This is the delay time clients transmit their results to the server.

Client service time may vary according to different subtasks but as an example the estimated parameter for queueing analytical model is indicated in table 5.

Once server, network and client service time are estimated then EZSIM simulation [6] will be employed. By applying a Poisson process for the arrival and Exponential distribution for service time, queueing model as shown in figure 3 will be constructed. All queue disciplines are set to be FCFS basis with unlimited size. The simulation will be running long enough to reach the steady state, as random number is part of the execution.

4.Results and Analysis

Gain (Speedup) is sequential processing time divided by parallel processing time. It is clearly seen that although our proposed model produces different results (lower) compared vis-à-vis results collected by 80 parallel computing units but the curve goes in the same direction (trend) as shown in figure 5.

5.CONCLUSION

With the identical (but lower) direction, the compensation can be predicted for our proposed model in order to approach an exact solution of the practical 80 units. The research to extend number of clients up to 200 is now undergoing in order to check the speed-up and overhead with results from parallel processing system. If cost-effectiveness is concerned then it can be concluded that the proposed method can fasten the process in receiving performance measures, which always go in the line of accurate results from simulation. Not to mention that computation cost is next to zero compared to time consumed by simulation. The appropriate number of processing units regarding to each executable tasks is also under the investigation for future research in order to lead to the optimization.

Clients	Sc	A	W	C	S
200	0.005941883	0.000000015	0.000000108	0.005941494	2.652E-07
160	0.007427353	1.875E-08	0.000000135	0.007426868	3.315E-07
120	0.009903138	0.000000025	0.00000018	0.009902491	0.000000442
80	0.014854706	3.75E-08	0.00000027	0.014853736	0.000000663
72	0.016505229	4.16667E-08	0.0000003	0.016504151	7.36667E-07
64	0.018568383	4.6875E-08	3.375E-07	0.01856717	8.2875E-07
56	0.021221009	5.35714E-08	3.85714E-07	0.021219623	9.47143E-07
48	0.024757844	6.25E-08	0.00000045	0.024756226	0.000001105
40	0.029709413	0.000000075	0.00000054	0.029707472	0.000001326
32	0.037136766	9.375E-08	0.000000675	0.03713434	1.6575E-06
24	0.049515688	0.000000125	0.0000009	0.049512453	0.00000221
16	0.074273532	1.875E-07	0.00000135	0.074268679	0.000003315
8	0.148547064	0.000000375	0.0000027	0.148537359	0.00000663

Table 5. Client service time (sec) for polynomial function 100^100.

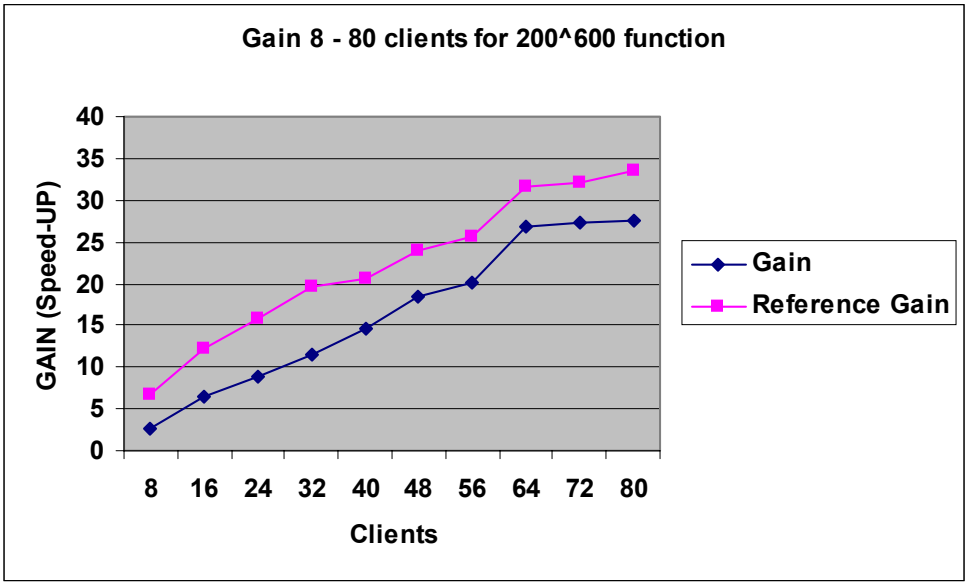

Figure 5. Gain 8 – 80 clients for 200^600 function.

6.References

[1] Fujimoto and Richard M., "Parallel and Distributed Simulation Systems.," 2000.

[2] Sutaweesup, Wasara, and Yuen Poovorawan. "Parallel Motion Path Calculation for Animated Objects in Distributed Environment," 2000.

[3] Kormicki, Maciek, Ausif Mahmood and Bradley S. Carlson, "Parallel Logic Simulation on a Network of Workstations Using a Parallel Virtual Machine," Washington State University at Tri-Cities, University of Bridgeport, State University of New York at Stony Brook, pp. 123-134, 1997.

[4] Wail Elias Mardini, "Modeling with D-queues," The University of New Brunswick, April, 2001.

[5] Donald Gross and Carl M. Harris. "Fundamentals of Queueing Theory,": third edition, Canada, A Wiley-Interscience Publication, 1998.

[6] Behrokh Khoshnevis. "Discrete systems simulation," University of southern California, McGraw-Hill inc., 1994.