Simulated Heterogeneous Processor Scheduling for Balanced Job Allocation

Harpreet Kaur, Ankit Arora, Gursharanjit cheema

Abstract: Parallel scheduling a task of utilizing multiprocessor hardware to formulate balanced system load and increased system viability. Policies to schedule parallel applications performs tremendous amount of distribution efforts in order to perform task placement and adjustment. Task adjustment basically a fine-tuning of system running state, where stabilizing load is the key factor to system increased throughput and consistent feasibility. Dynamic scheduling, an ability to evaluate processor characteristics and workload characterization, an ongoing measurement process to consistent workload distribution. This research towards processor frequency measurements, considered to make balanced load distribution, rather than consuming efforts for task adjustment later after distribution. Cycle based metric provides measurement as no. of cycles spent in one unit time by the processor. In heterogeneous processor interconnection/organization where this measurement will carries a great aspect of real-life parallel application characterization. Further the research will carries a heterogeneous multiprocessors typically a single core simulated job assignment with frequency estimation.

Keywords: Balanced Job Allocation, Heterogeneous Multiprocessor Simulation, Load Consistency, Processor Cycle Speed,

I. INTRODUCTION

Parallel processors usually based on single core homogeneous system but in reality high complex applications has varied amount of load and processing modules i.e. their capabilities and capacities are different. Therefore mapping of such real applications over homogeneous parallel interconnection is a static aspect of distribution. Each application controlled by any available processor regardless of measuring their underlying capacity and task size. Load distribution among homogeneous interconnection system will not get benefit of workload characterization, so heterogeneous interconnection system provides a way to stabilize load among processor's efficiency and capacity corresponds to task requirements [1]. Processor efficiency measurements basically a field under which processor stress will be estimated, after a long duration due to complex data processing or computation, the efficiency may degrades. This is the point at which load balancing will takes place. Under this scheme, the idea is to distribute the load of high stressed processors over stress free processors. These methods will carry huge amount of efforts because at level one the load will be distributed and later at level two the load will be adjusted, so large no. of context switching operations will be the result. In this scheme, the efforts of load adjustment will be reduced. Concentration is on balanced load assignment during job distribution rather than level two process, which is not required. Due to unbalanced load scheduling, jobs are distributed among numerous processors, also improper consumption of resources. Maximum effort should have to perform for fine-tuning the overall operations with minimum change in process address space, very tricky aspect of OS scheduling that consume number of processor's cycles. Further Multicore processor's can be programmed via this dynamic factorization. The core can be further of homogeneous speed or they can be heterogeneous, to cover the intermediate load balancing .so, heterogeneous clusters can be organized to make cloud oriented services where applications demands a particular set of resources. That's why the demand for increased heterogeneity in computing systems is reasonably due to the need for high-performance, highly reactive systems that interact with other environments (audio/video systems, control systems, networked applications) etc.

Harpreet Kaur, a student of engineering presents this material on the behalf of research thesis Email: hbuttar55@yahoo.com. Ankit Arora, Assistant Professor in LLRIET, Moga Pb India, internal thesis guide, Email:ankitrajan11@yahoo.com. Gursharanjit cheema, Assistant Professor LLRIET, Moga Pb. India provides his well known cooperative assistance to thesis work, Email: cheema_303@yahoo.com.

II. OBJECTIVE

Objective under processor frequency estimation is to perform task distribution according to processor cycle speed and processing load. In other words, measuring existing load of each processor in the communication along with their speed and finds intended processor, which is a key component of new task assignment. Key-Processor, which requires minimum no. of cycles for its existing and new workload ready to be assigned. This processor will be more efficient and effective in current scenario. The main aim of this research is to balance the workload over the processors. Load will be balanced via proper distribution of jobs to the processors. Unbalanced load assignment refers that some processors have vast amount of processing deeds and other processors may become idle. In general processor load will define the processor stress, managing stress in multiprocessor is a critical task. Stress can be managed during allocation, a ongoing process that will measure the processor stress in terms of load given and assign new workload to it if having capacity to adapt. So task placement if performed accurately ultimately the stress will be balanced among processors. Other wise, task requires reallocation.

III. LITERATURE REVIEW

Many of the existing literatures implements static multi-processor scheduling covering predefined parametric factors [2]. Other literature describes moldable and malleable demand allocation where processor demands are adjusted to current processor availability. Literature around multi-computer cluster interconnection analyzes behavior of parallel algorithms with divide and conquer approaches. Such literatures illustrates several difficulties occurred around network data transmission and communication delays. Synchronization will be crucial aspect in network clustering. Scheduling over heterogeneous processor plays critical role in modern real applications where each task may demand different set of resource requirements in parallel execution interval. Capabilities are varying, load balancing with dynamic job allocation policies along with workload characterization is an essential part of multiprocessor distribution. Other research related with parallel processing is bounded buffered scheduling schemes along with homogeneous interconnection. Processor availability width is mapped to frequently arrived jobs along with limited buffer space with the aim of increased throughput. Current research takes care of dynamic scheduling around processor frequency estimation with the aim of consistent load allocation. This method can be further analyzed with core processor technology in heterogeneous multi-core multi-processor systems. Simulation theory around previous research shows logically programmed multiprocessor scheduling arrangement having n number of processors and m number of jobs. The scheduler allocates multiple jobs to single processor or single job to multiple processor. Each job has its CPU burst cycle. Improper distribution of job lead to the unbalanced load among processors

IV. ARCHITECTURAL LAYOUT

Architectural layout behind processor interconnection incorporates heterogeneous multi-processors along with discrete frequency speed. The application varying length of modules can be easily mapped with heterogeneous structures. Following is the general structure of underlying processor architecture. Simulation structure follows logical layout described in fig 1 for processor interconnection where random distribution will be used for workload generation. The generated workload is then distributed to the processor with frequency estimation metric by measuring existing processor load. Synchronized multithreading environment is created in visual basic 6.0 programming language. Periodically, existing load will be estimated during each job assignment and processor index will be computed to which the current assignment must take place .

V. ALGORITHMIC FLOW

Algorithmic structure follows cycle speed estimation for each processor i.e. no. of cycles or capability of cycles that the processor has elapsed in one second. Fig 2 describes complete control flow chart. This algorithm is completely different from other algorithmic structures incorporates static behavior etc.

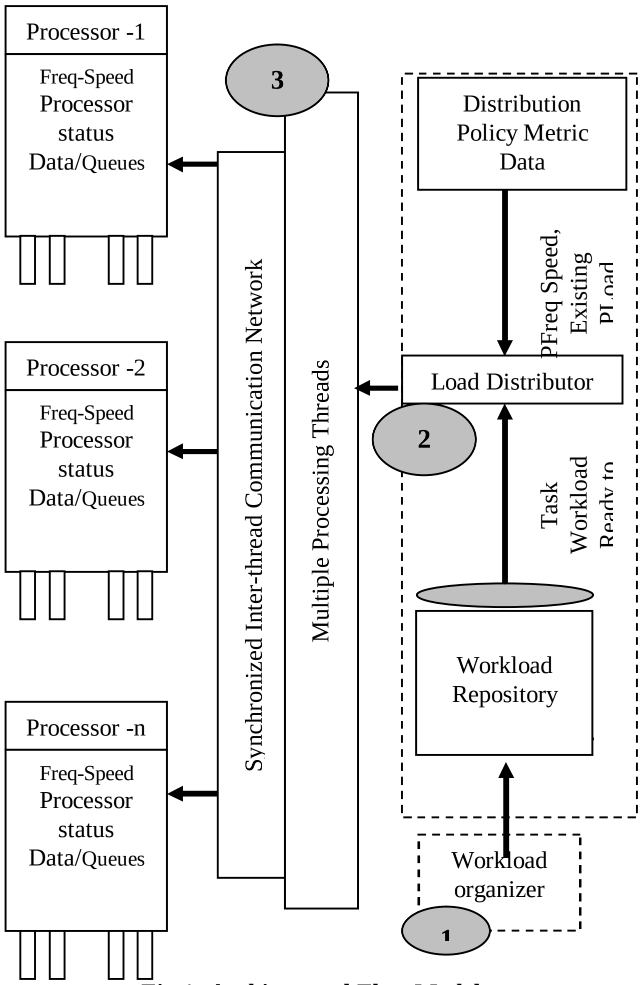

Processor -1
Freq-Speed Processor status Data/Queues
Processor -2
Freq-Speed Processor status Data/Queues
Processor -n
Freq-Speed Processor status Data/Queues
3
Synchronized Inter-thread Communication Network
Multiple Processing Threads
Distribution Policy Metric Data
PFreq Speed, Existing PLoad
Load Distributor
2
Task Workload Ready to
Workload Repository
Workload organizer
1

Fig 1: Architectural Flow Model

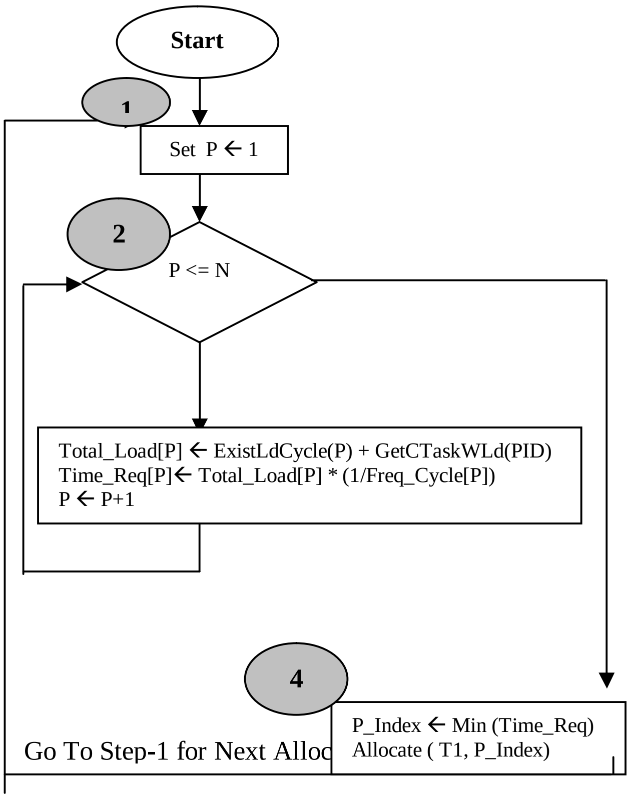

Start
1
Set P ← 1
2
P <= N
Total_Load[P] ← ExistLdCycle(P) + GetCTaskWLd(PID)
Time_Req[P] ← Total_Load[P] * (1/Freq_Cycle[P])
P ← P+1
4
P_Index ← Min (Time_Req)
Allocate (T1, P_Index)
Go To Step-1 for Next Alloc

Fig 2: Algorithmic Flow

N is the number of processors in the communication. Each P^th^ processor has existing load in terms of tasks assigned and each task has CPU burst cycle. For N processors the complexity order will be O(N). For each P^th^ processor complexity metric for existing load computation is–

$$Eload\ [P] = \left[\sum_{i=1}^{Queue\_Length[P]} TaskWLoadCycle[P][i] \right] \quad (1)$$

$$Tload\ [P] = Eload\ [P] + CTaskWLoad \quad (2)$$

$$T\_Req\ [P] = \sum_{P=1}^{N} \left[Eload[P] * \frac{1}{Freq\_Cycle[P]} \right] \quad (3)$$

VI. LOAD ADJUSTMENT POLICIES

The approach above defined provides many benefits than load adjustment policies. Adjustment policies basically takes care of tasks previously assigned, In this tasks are reassigned to processors via some load balancing schemes includes efforts to balance load among them rather than performing any computation intensive work [3][4]. Despite of this one another approach which considers dynamic processor characteristic and workload characterization during workload assignment rather than readjusting later at unbalancing check-points [5]. In the previous one statically load is distributed and later steadiness will be provided when required with the aim that the stage for load balancing when happen only then the recovery schemes will be adopted rather than incorporating additional efforts from the very beginning when execution starts. Parallel processing, where vast amount of computation and data intensive applications arrived then such situations are happened very frequently and requires immediate prevention schemes to maintain system steady state [6]. Despite of load adjustment to balance system steady state after load allocation, stress management policies may be adapted for multiprocessors. Processor efficiency measurement will provide dynamic aspect to enhance performance degradation Processor utilization with in one particular duration period will inspire to manage load balancing issues. If the load distribution is performed after computing processor stress then it will be a balanced load allocation and does not requires task adjustment plans.

VII. RESULTS AND DISCUSSIONS

Simulation Results predicated is the outcome of different running scenarios integrating random workload. Illustration produced exhibits steadiness in workload distribution is performed up to very large extent. The samples collected will be based upon different time barrier points. These points are basically the timing checkpoints where simulation working is stopped to get current status of the running scenario. The simulated view is described further containing fifteen heterogeneous processors asynchronously behaved like MIMD processors. Each processor has its own working clock frequency by which the load will be distributed as described above in the literature.

I. Simulation Status at Time-25

Sr. No.	Frequency	Overall Load cycle	No. of Jobs
1	100 MHZ	0	0
2	500 MHZ	0	0
3	1 GHZ	0	0
4	1.6 GHZ	1537	1
5	2.2 GHZ	3406	1
6	2.7 GHZ	3713	2
7	3.2 GHZ	2827	1
8	3.7 GHZ	4927	1
9	4.2 GHZ	4803	2
10	4.8 GHZ	7649	3
11	5.4 GHZ	9387	2
12	6 GHZ	12809	3
13	6.5 GHZ	9344	2
14	7 GHZ	11485	3
15	7.6 GHZ	15849	4

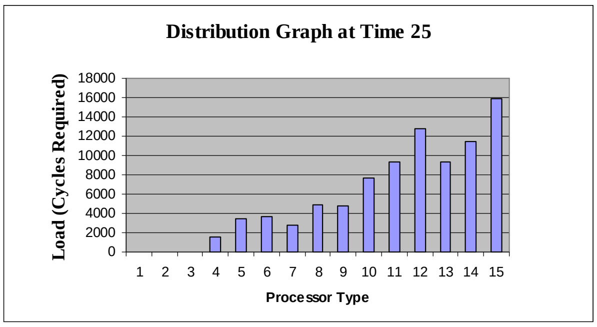


Fig 3: Distribution Graph-1

II. Simulation Status at Time-50

Sr. No.	Frequency	Overall Load cycle	No. of Jobs
1	100 MHZ	0	0
2	500 MHZ	0	0
3	1 GHZ	1704	1
4	1.6 GHZ	2323	2
5	2.2 GHZ	4954	3
6	2.7 GHZ	6109	3
7	3.2 GHZ	6888	4
8	3.7 GHZ	7942	3
9	4.2 GHZ	8958	4
10	4.8 GHZ	13153	5
11	5.4 GHZ	13198	4
12	6 GHZ	15987	5
13	6.5 GHZ	16187	5
14	7 GHZ	16373	5
15	7.6 GHZ	21934	6

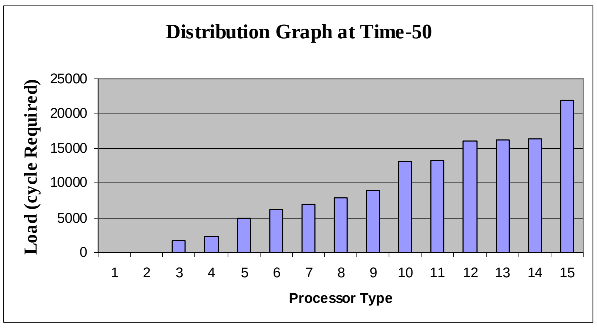


Fig 4: Distribution Graph-2

III. Simulation Status at Time-100

Sr. No.	Frequency	Overall Load cycle	No. of Jobs
1	100 MHZ	0	0
2	500 MHZ	1541	1
3	1 GHZ	2645	3
4	1.6 GHZ	4447	5
5	2.2 GHZ	7349	6
6	2.7 GHZ	9212	7
7	3.2 GHZ	10528	8
8	3.7 GHZ	13112	8
9	4.2 GHZ	13115	8
10	4.8 GHZ	19206	8
11	5.4 GHZ	19637	9
12	6 GHZ	21223	9
13	6.5 GHZ	22676	9
14	7 GHZ	24660	9
15	7.6 GHZ	29891	10

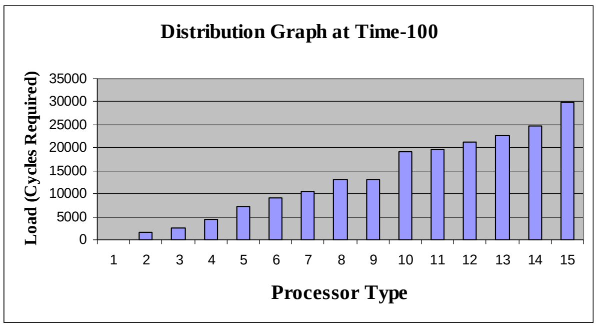


Fig 5: Distribution Graph-3

IV. Simulation Status at Time-150

Sr. No.	Frequency	Overall Load cycle	No. of Jobs
1	100 MHZ	0	0
2	500 MHZ	1589	3
3	1 GHZ	4106	6
4	1.6 GHZ	6786	7
5	2.2 GHZ	10578	10
6	2.7 GHZ	14441	10
7	3.2 GHZ	16559	12
8	3.7 GHZ	17135	12
9	4.2 GHZ	22462	12
10	4.8 GHZ	24219	11
11	5.4 GHZ	26412	13
12	6 GHZ	29901	13
13	6.5 GHZ	35616	13
14	7 GHZ	38548	14
15	7.6 GHZ	38671	14

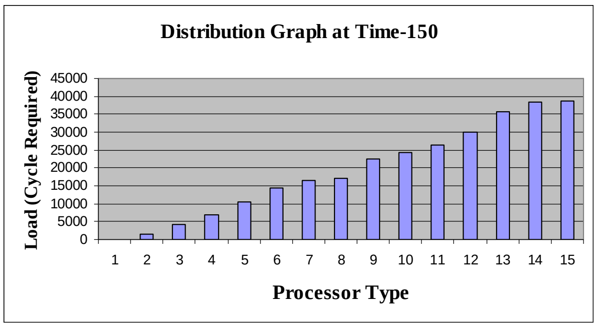


Fig 6: Distribution Graph-4

Sr. No.	Frequency	Overall Load cycle	No. of Jobs
1	100 MHZ	0	0
2	500 MHZ	2338	6
3	1 GHZ	4837	8
4	1.6 GHZ	8479	10
5	2.2 GHZ	14451	13
6	2.7 GHZ	15774	14
7	3.2 GHZ	20942	15
8	3.7 GHZ	24303	16
9	4.2 GHZ	27901	16
10	4.8 GHZ	29235	14
11	5.4 GHZ	34093	18
12	6 GHZ	40245	17
13	6.5 GHZ	43800	17
14	7 GHZ	47216	18
15	7.6 GHZ	47807	18

V. Simulation Status at Time-200

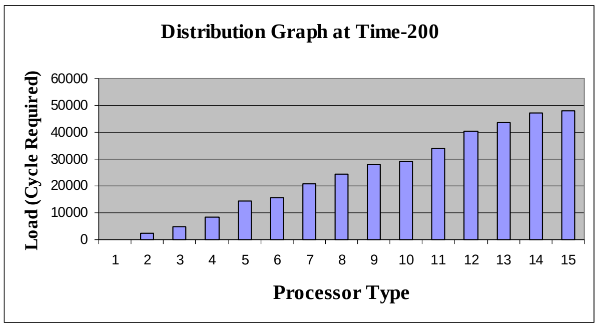


Fig 7: Distribution Graph-5

VI. Simulation Status at Time-350

Sr. No.	Frequency	Overall Load cycle	No. of Jobs
1	100 MHZ	0	0
2	500 MHZ	3905	13
3	1 GHZ	9288	17
4	1.6 GHZ	15211	22
5	2.2 GHZ	20150	22
6	2.7 GHZ	24562	25
7	3.2 GHZ	29998	24
8	3.7 GHZ	38082	26
9	4.2 GHZ	41231	26
10	4.8 GHZ	46142	26
11	5.4 GHZ	56102	29
12	6 GHZ	61380	28
13	6.5 GHZ	65039	30
14	7 GHZ	68997	30
15	7.6 GHZ	75415	32

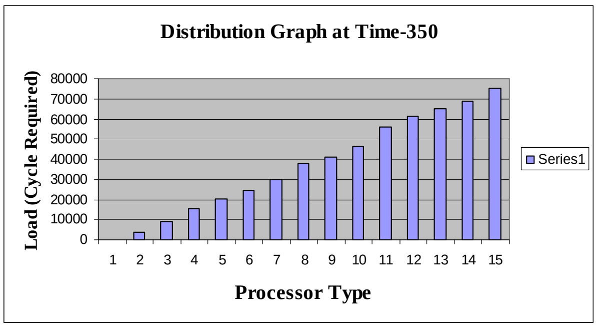


Fig 8: Distribution Graph-6

VII. Simulation Status at Time-475

Sr. No.	Frequency	Overall Load cycle	No. of Jobs
1	100 MHZ	1008	2
2	500 MHZ	5180	18
3	1 GHZ	11734	24
4	1.6 GHZ	19068	30
5	2.2 GHZ	26299	29
6	2.7 GHZ	33205	35
7	3.2 GHZ	38596	30
8	3.7 GHZ	44916	35
9	4.2 GHZ	52419	34
10	4.8 GHZ	61733	37
11	5.4 GHZ	67476	36
12	6 GHZ	75755	39
13	6.5 GHZ	85278	41
14	7 GHZ	89914	42
15	7.6 GHZ	98199	43

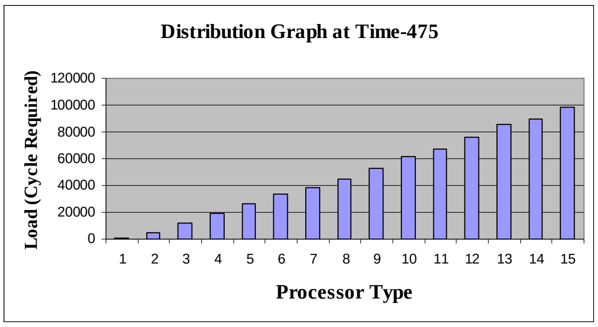


Fig 9: Distribution Graph-7

VIII. CONCLUSION & FUTURE WORK

Results concluded so far incorporates asynchronously arranged heterogeneous processors by taking long duration execution samples illustrates from the very beginning load is unstable but as the simulation proceeds further, increased improvement over distribution is the result, this is because initially from the very beginning the processor are lightly loaded and stableness and unstableness will be best described when system revolves around heavy load, only then load stability metrics will be measured and justified This method of load distribution performs balanced load assignment without requiring task adjustment efforts. Very long run execution of simulation provides a way to characterize workload consistently. Processor frequency a great factor against load balancing around heterogeneous multiprocessor interconnection. Randomly task workload is generated containing burst cycles are then computed with frequency metric and distributed. Further the future work may incorporate other dynamic factors like FLOPS (floating point operations per second), MIPS (million instructions per seconds) these all are dependent upon the processor frequency of execution. Each processor has varying execution speed which in turn leads to execution of varying no. of instructions per second. Such types of dynamic factor for workload characterization will results in increased performance and throughput

IX. References

[1] Bobrek A., Paul M. IEEE Stochastic Contention for Single-Chip Heterogeneous Multiprocessor. Vol. 59, pp. 1402-1418 ,Oct 2010

[2] Arora, S., Arora, A. Scheduling simulations: An experimental approach to time-sharing multiprocessor scheduling schemes. Foundation of Computer Science New York. vol. 63, pp. 29-35, feb 2013.

[3] Marc, H. Lemair, W.. Strategies for load balancing for highly parallel computers IEEE transactions on parallel and distributed systems Vol. 4, pp. 979-993, Sep.1993

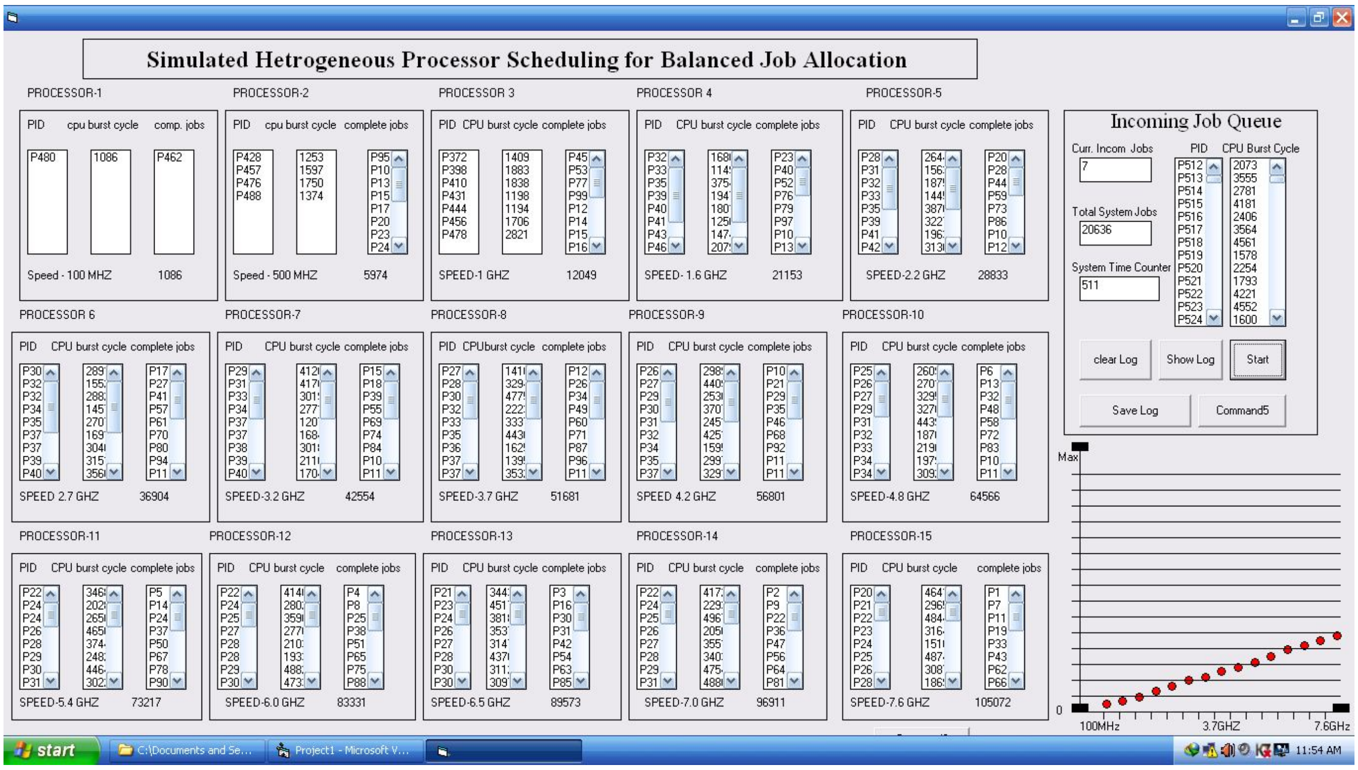
[4] Jacques, M. and Couturier, R. IEEE, Sylvain Contassot-Vivier, Member, Dynamic Load Balancing and Efficient Load Estimators for Asynchronous Iterative Algorithms vol. 16, pp. 289-299, Apr 2005

[5] Chhabra, A. and Singh, G. Simulated Performance Analysis of Multiprocessor Dynamic Space-Sharing Scheduling policy vol. 9, pp. 326-329, 2009

[6] Cybenko, G. 1989. Dynamic load balancing for distributed memory multi-processor Department of Computer Science, Tufts University, Medford, Massachusetts. vol. 7 pp. 279-301, Feb 1989

[7] Mendel R, Complete Computer System Simulation:The SimOS approach IEEE Parallel and Distributed Computing pp. 34-43, 1995

[8] Rudloph L, A Simple Load Balancing scheme for task allocation in Parallel Machines ACM, pp. 237-245, 1995



Simulated Hetrogeneous Processor Scheduling for Balanced Job Allocation

PROCESSOR-1

PID	cpu burst cycle	comp. jobs
P480	1086	P462

Speed - 100 MHZ 1086

PROCESSOR-2

PID	cpu burst cycle	complete jobs
P428	1253	P95
P457	1597	P10
P476	1750	P13
P488	1374	P15
		P17
		P20
		P23
		P24

Speed - 500 MHZ 5974

PROCESSOR 3

PID	CPU burst cycle	complete jobs
P372	1409	P45
P398	1883	P53
P410	1838	P77
P431	1198	P99
P444	1194	P12
P456	1706	P14
P478	2821	P15
		P16

SPEED-1 GHZ 12049

PROCESSOR 4

PID	CPU burst cycle	complete jobs
P32	168	P23
P33	114	P40
P35	375	P52
P39	194	P76
P40	180	P79
P41	125	P97
P43	147	P10
P46	207	P13

SPEED- 1.6 GHZ 21153

PROCESSOR-5

PID	CPU burst cycle	complete jobs
P28	264	P20
P31	156	P28
P32	187	P44
P33	144	P59
P35	387	P73
P39	322	P86
P41	196	P10
P42	313	P12

SPEED-2.2 GHZ 28833

PROCESSOR 6

PID	CPU burst cycle	complete jobs
P30	289	P17
P32	155	P27
P32	288	P41
P34	145	P57
P35	270	P61
P37	169	P70
P37	304	P80
P39	315	P94
P40	356	P11

SPEED 2.7 GHZ 36904

PROCESSOR-7

PID	CPU burst cycle	complete jobs
P29	412	P15
P31	417	P18
P33	301	P39
P34	277	P55
P37	120	P69
P37	168	P74
P38	301	P84
P39	211	P10
P40	170	P11

SPEED-3.2 GHZ 42554

PROCESSOR-8

PID	CPUburst cycle	complete jobs
P27	141	P12
P28	329	P26
P30	477	P34
P32	222	P49
P33	333	P60
P35	443	P71
P36	162	P87
P37	139	P96
P37	353	P11

SPEED-3.7 GHZ 51681

PROCESSOR-9

PID	CPU burst cycle	complete jobs
P26	298	P10
P27	440	P21
P29	253	P29
P30	370	P35
P31	245	P46
P32	425	P68
P34	159	P92
P35	299	P11
P37	329	P11

SPEED 4.2 GHZ 56801

PROCESSOR-10

PID	CPU burst cycle	complete jobs
P25	260	P6
P26	270	P13
P27	329	P32
P29	327	P48
P31	443	P58
P32	187	P72
P33	219	P83
P34	197	P10
P34	309	P11

SPEED-4.8 GHZ 64566

PROCESSOR-11

PID	CPU burst cycle	complete jobs
P22	346	P5
P24	202	P14
P24	265	P24
P26	465	P37
P28	374	P50
P29	248	P67
P30	446	P78
P31	302	P90

SPEED-5.4 GHZ 73217

PROCESSOR-12

PID	CPU burst cycle	complete jobs
P22	414	P4
P24	280	P8
P25	359	P25
P27	277	P38
P28	210	P51
P28	193	P65
P29	488	P75
P30	473	P88

SPEED-6.0 GHZ 83331

PROCESSOR-13

PID	CPU burst cycle	complete jobs
P21	344	P3
P23	451	P16
P24	381	P30
P26	353	P31
P27	314	P42
P28	437	P54
P30	311	P63
P30	309	P85

SPEED-6.5 GHZ 89573

PROCESSOR-14

PID	CPU burst cycle	complete jobs
P22	417	P2
P24	229	P9
P25	496	P22
P26	205	P36
P27	355	P47
P28	340	P56
P29	475	P64
P31	488	P81

SPEED-7.0 GHZ 96911

PROCESSOR-15

PID	CPU burst cycle	complete jobs
P20	464	P1
P21	296	P7
P22	484	P11
P23	316	P19
P24	151	P33
P25	487	P43
P26	308	P62
P28	186	P66

SPEED-7.6 GHZ 105072

Incoming Job Queue

Curr. Incom Jobs: 7

Total System Jobs: 20636

System Time Counter: 511

PID	CPU Burst Cycle
P512	2073
P513	3555
P514	2781
P515	4181
P516	2406
P517	3564
P518	4561
P519	1578
P520	2254
P521	1793
P522	4221
P523	4552
P524	1600

Fig 10: Simulated view