High Performance Model for Handling Machine Breakdown in Identical Parallel Machines

Onwuachu Uzochukwu C.
Department of Computer Science,
Imo State University,
Owerri, Imo State, Nigeria

Ugwu C.
Department of Computer Sciences,
University of Port Harcourt,
Choba, Rivers State, Nigeria

Williams Edem
Department of Computer Science,
University of Calabar,
Cross River State, Nigeria

ABSTRACT
Machine breakdown is an issue that cannot be overlooked when considering the performance of any scheduling model. This issue has resulted in the inability to meet up with the job due date and also increases job completion time among identical parallel machines. Therefore, an efficient job scheduling model will take care of machine failure issues to obtain a good job schedule. This paper developed an efficient scheduling model that is robust, to handle the issues of machine failure and minimize the total completion time for job execution in identical parallel machines. The developed model adopted fuzzy logic technique in developing a job dispatcher for the identical parallel machines. The job dispatcher was used in determining the available machine and the failed machine before dispatching jobs to the individual parallel machines. The model was tested with fifteen identical parallel machines used for printing jobs in the printing press. The parameter used in analyzing this model includes the machine load balancing and machine utilization. The result from this model was compared with other existing model like first come first serve scheduling model and genetic scheduling model. The lowest machine utilization recorded from the experiment conducted using first come first serve scheduling model, genetic scheduling model and the developed scheduling model was 83.46856%, 89.57643% and 98.2949% respectively, which shows that the new model achieved better load balancing and efficient machine utilization among the identical parallel machines.

Keywords
Job scheduling model, machine breakdown, machine utilization, identical parallel machines and load balancing.

1. INTRODUCTION
Job scheduling problem subject to machine breakdown is one of the challenging issues in production field. Robustness and stability are the important measures to consider when we reschedule jobs [1]. Machine breakdown have lead to so many losses in production industries and one of its effect is the inability to meet the job due date. The challenges of job scheduling in an identical parallel machine are dealing with the computing resources for the number of jobs, considering the following factors which include complexity, dependency, resource starvation, load balancing and efficiency [2]. Utilizing dispatching rules is a well-known technique for taking care of scheduling issues. The dispatching rules figures out what job to process straightaway. The scheduling problem deals with the optimal assignment of jobs to the identical parallel machine and orders their execution so that the total completion time is minimized [3].

Identical parallel machines are important resources that are generally shared by communities of users. The charge of job scheduling is to decide when and how each job should be carried out in order to exploit the system's cumulative value to its owners [4]. The way in which jobs are allocated to machine is fundamental to realizing the high performance of the corresponding systems, such as minimizing mean job response time and maximizing machine throughput. [5]

In general, the aim of job scheduling is to have load balancing among the identical parallel machines, whereas for the later minimization of overall execution time is the main concern [6]. The intention behind scheduling is to exploit the system's throughput by effecting maximum number of jobs in the given time span. Load balance [7] is considered as a major problem when scheduling multiple jobs with limited resources. The load balance should be minimized to improve the machine throughput and efficiency.

Heuristic optimization algorithm [7] is broadly used to solve a diversity of problems. [8]Abraham and Nath (2000) proposed three basic heuristics implied in nature for grid scheduling, namely Genetic Algorithm [9], Simulated Annealing [10] and Tabu Search [11], and heuristics derived by a combination of these three algorithms are very powerful.

The job scheduling algorithm is considered a complex process because it must schedule a large number of jobs into the available resources [12][13]. This paper identifies a number of issues that may be experienced when utilizing the identical machine in the scenario of job scheduling, namely mapping, arrangement of execution, and optimal configuration of the identical parallel machines. [14] There is a need therefore to develop a model for job dispatch to conquer these issues. [15] The enabling strategy is the use of job scheduling plan based on genetic algorithm and fuzzy logic to determine these issues.

2. LITERATURE REVIEW
Savas (2012) worked on non-identical parallel machine scheduling with fuzzy processing times using robust genetic algorithm and simulation. His research addresses non-identical parallel machine scheduling problem with fuzzy processing times (FPMSP). The proposed GA approach yields good results and can explore alternative schedules providing the same results.[16]

Mostafa et al. (2012) searched for an optimal solution in scheduling real-world problems in industrial applications, especially for mission time critical systems. A parallel GA was employed to solve flow shop scheduling issues with the aim of minimizing the makespan. It was found that the proposed parallel genetic algorithm (PPGA) considerably

decreased the CPU time without adversely affecting the makespan.[17]

Rachhpal (2012) worked on task scheduling with genetic approach and task duplication technique. In his paper, numbers of alternative solutions and heuristics techniques were proposed to solve the problem. He noticed that applying the GA with task duplication technique enhances the efficiency of the task scheduling in parallel multiprocessor environment.[18]

Kaleeswaran et al (2012) worked on dynamic scheduling of data using genetic algorithm in cloud computing. They discovered that task arrival was uncertain at run time in dynamic scheduling and allocating resources was tedious as several tasks arrive at the same time. Their result reduced the execution time in parallel processing and obtained global optimization.[19]

Zubair et al. (2012) worked on tasks allocation using fuzzy inference in parallel and distributed system. They proposed a task assignment model and multi-agent distributed approach based on fuzzy assessment of machines and a virtual budget to assign tasks to processing element in a dynamic environment. Parallel and distributed computations were performed directly within the interpretive MATLAB environment. They found some good response time of tasks in parallel and distributed system.[20]

Ali (2012) worked on fuzzy dynamic load balancing algorithm for homogenous distributed systems. He proposed a new fuzzy dynamic load balancing algorithm for homogenous distributed systems. The proposed algorithm utilizes fuzzy logic in dealing with inaccurate load information, making load distribution decisions, and maintaining overall system stability. In terms of control, they proposed a new approach that specifies how, when, and by which node the load balancing is implemented. [21]

Mohammad and Mehdi (2013) worked on high performance scheduling in parallel heterogeneous multiprocessor systems using evolutionary algorithms. In their research, they introduced a method based on genetic algorithms for scheduling and load balancing in parallel heterogeneous multi-processor systems. The results of the simulations indicated that Genetic algorithm is better than LPT, SPT and FIFO. Simulation results indicate Genetic Algorithm reduces total response time and also it increases machine utilization. [22]

Prabhjot and Amanpreet (2013) worked on implementation of dynamic level scheduling algorithm using genetic operators. In their research they implemented APN Dynamic Level Scheduling algorithm by using genetic operators for task scheduling in parallel multiprocessor system including the communication delays to reduce the completion time and to increase the throughput of the system. The parameters used are makespan time, processor utilization and scheduled length ratio. The graphs showed better results of dynamic level scheduling with genetic operators as compared to simple dynamic level scheduling algorithm.[23]

Aparna et al (2014) worked on task scheduling in homogeneous multiprocessor systems using evolutionary techniques. Their research shows how genetic algorithms can be adapted to tackle this problem, considering first a single task network and then a number of task networks with a given time period. For the single network as well as for a group of networks treated independently in a multi-network system, the research shows that the average of total processing time decreases with respect to generations. [24]

Leila (2014) worked on solving the Job Shop Scheduling problem with a Parallel and Agent-Based Local Search Genetic Algorithm. In his research, he presented a parallel and agent-based local search genetic algorithm for solving the job shop scheduling problem. A multi-agent system containing various agents each with special behavior is developed to implement the parallel local search genetic algorithm. Benchmark instances are used to investigate the performance of the proposed approach. The results show that the proposed agent-based parallel local search genetic algorithm improves the efficiency.[25]

Selvi (2014) worked on multi objective optimization problems on identical parallel machine scheduling using genetic algorithms. He attempted to solve scheduling problems involving identical parallel machines, where the objective is to optimize the multi-objective scheduling problems using Genetic algorithms. The major contribution of present work lies in proposing mathematical models using Genetic algorithms to optimize major objectives and to study the effectiveness of these algorithms for small and large size problems.[26]

Zeinab and Seyed (2014) worked on novel decentralized fuzzy-based approach for grid job. They followed the identification of grid scheduling with the help of fuzzy theory and sought to present a new method for grid scheduling with respect to exiting obstacles. The results of the experiments show the efficiency of the proposed method in terms of makespan and standard deviation of the load of clusters. [27]

Rachhpal (2014) worked on task scheduling in parallel systems using genetic algorithm. In his work, genetic algorithm based on the principles of evolution to obtain an optimal solution for task scheduling is developed. Genetic algorithm was based on three operators: Natural Selection, Crossover and Mutation. The simulation results prove that the method proposed generates better results. [28]

Seyed et al. (2015) looked at fuzzy genetic algorithm for scheduling of handling/storage equipment in automated container terminals. They designed a Fuzzy Logic Controller (FLC) to improve the performance of a GA in optimization of integrated scheduling of handling/storage equipment in automated container terminals. The FLC controls crossover and mutation rates of the GA during its generations, which are the main control parameters of the GA to avoid the premature convergence. The numerical results for the small size test cases solved by using the proposed fuzzy genetic algorithm showed that solutions found by this algorithm are 2.5% better than the solutions found by the GA. [29]

3. MATERIALS AND METHODS

In order to solve the issue of complexity in handling machine failure faced by the existing system, we proposed a fuzzy model for scheduling job among identical parallel machine systems. The proposed system has a set of n jobs, N = {1, 2, ... n} to be processed on m parallel machines. M = {1,2, ..., m} within a time window [ri, di]. A job j will be processed by the suitable machine m. All the machines have the same speed (V_i) and can process only one job at a time. The processing time of job j on machine m is denoted by P(m, j). The model is designed to minimize the total tardiness time and total completion time (makespan), also to give attention to jobs with early due date. The new system boolean variable X (i, j) which determines whether job j is processed by machine i(if

x(i, j) = 1) or not (if x(i, j) = 0). The system works efficiently based on the assumptions listed in 3.1

3.1 Principal Assumptions

In order to minimize the total tardiness time of scheduling grouped jobs in the identical parallel machines, the optimal solution satisfies the following conditions:

1. No machine can process more than one operation at a time.
2. Each operation, once started, must be performed till completion except the case of machine failure issue.
3. A job is an entity, *i.e.* Job cannot represent many individual parts, and will not be processed by more than one machine at a time.
4. Each operation must be completed before any other operation can begin.
5. Time intervals for processing are independent of the order in which operations are performed.
6. There can be only the same type of machine for the processing.
7. A job is processed as soon as possible subject to ordering requirements.
8. All jobs are known and are ready to start processing before the period under consideration begins.
9. The time required to transfer jobs between machines is negligible.

3.2 Determining machine failure and availability

The system uses these assumptions listed in 3.1 to develop a fuzzy model for job dispatch in identical parallel machines. The job dispatcher performs fuzzy reasoning jobs considering the job waiting time and machine availability in order to know which job to be dispatched to the individual machines. The fuzzification module receives the crisp numeric values as input, process them and map them into fuzzy membership function values. The fuzzy engine is responsible for processing all determined membership function values using fuzzy sets', with fuzzy rule base to identify the most suitable fuzzy output. However, the defuzzification module is responsible for converting the fuzzy output into a numeric output suitable for the environment decision and control situation. The fuzzy inference system determines the membership function for the job waiting time and machine availability which will be used in job allocation to the identical parallel systems.

The fuzzy sets for job waiting time depends on how long a particular job has to wait to be dispatched , the waiting time fuzzy values are giving as Short, Average, and Long. The machine availability depends on the state of the processing machines; machine availability fuzzy values are given as free, busy and failed.

3.2.1 Fuzzy Input/Output Specifications

The fuzzy input/output variables are used for job dispatching and it is assumed to be the base for the dispatching rule. The fuzzy I/O specification is presented in Table 1. There are two input variables and one output. The output is the effect of the result of input states.

3.2.2 Fuzzification

Fuzzification is the process of transforming a crisp value into a fuzzy set so that it can be used and processed by fuzzy inference mechanism. The inputs and output of the design as specified in Table 1 are assigned linguistic variables and some degrees of membership. For input one (the job waiting time), the corresponding range is short (0.0, 0.3), average (0.4, 0.6) and long (0.7, 1.0). For input two (machine availability), the corresponding value is failed = 0.0, many free = 0.25, two free = 0.5, one free = 0.75 and all busy = 1.0. The levels for output (job allocation) are retrieve job and send to next available device = 0.0, wait = 0.5, and allocate = 1.0. The fuzzy variables for this proposed model include the following:

Machine availability **(MA) = {**One Free (OF), Two Free (TF), Many Free (MF), All Busy (AB), Failed (F)}.

Waiting Time (WT) = {Short (S), Average (A), Long (L)}.

Job Allocation (JA) = {Allocate (A), Wait (W), Retrieve and Send to Available Machine (RSAM)}

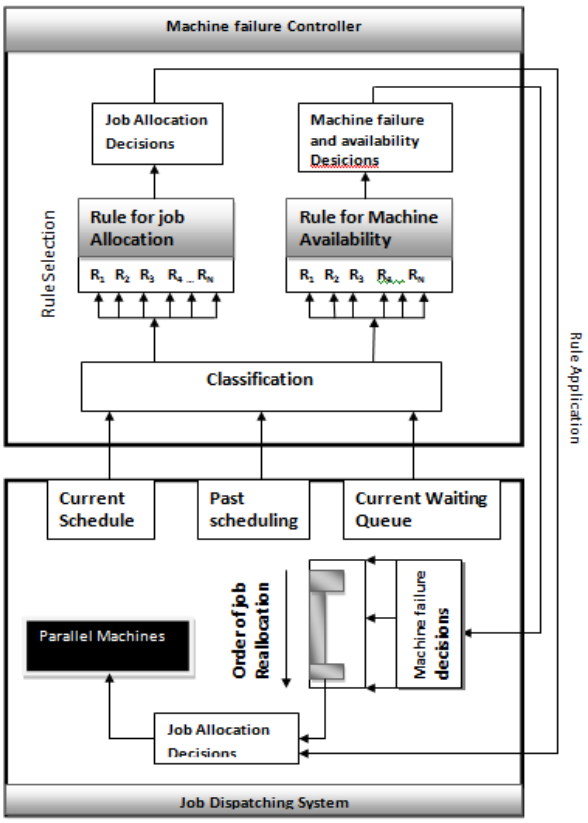

Figure 1. Job dispatching system.

Figure 1 shows the diagrammatic representation of job dispatching system. It handles the current schedule, the past scheduling and current waiting queue. Here we have two different rule selections, which include the rules for job allocation and rules for machine availability. These rules are used to take decisions on job allocation, machine failure, and machine availability decisions. In machine failure decisions, the unexecuted job will be reallocated to the next available machine. Job allocation decisions are used in allocating jobs to the parallel machine.

Table 1: Fuzzy variables (Input/Output Specifications)

INPUT		OUTPUT
Machine availability (MA)	**Waiting Time (WT)**	**Job Allocation (JA)**
One Free (OF) Two Free (TF) Many Free (MF) All Busy (AB) Failed (F)	Short (S) Average (A) Long (L)	Allocate (A) Wait (W) Retrieve and Send to Available Machine (RSAM)

3.2.3 Fuzzy Rules for machine availability.

The rules are formulated using a series of if-then statements, combined with AND/OR operators. For instance, if one device is free AND the job waiting time is small THEN job allocation should wait. With two inputs having eight membership functions, we have 5 * 3 = 15 rules as showed in table 2.

R1: If MA = OF, and WT = S then JA = W

R2: If MA = OF, and WT = A then JA = W

R3: If MA = OF, and WT = L then JA = A

R4: If MA = TF, and WT = S then JA = W

R5: If MA = TF, and WT = A then JA = A

R6: If MA = TF, and WT = L then JA = A

R7: If MA = MF, and WT = S then JA = A

R8: If MA = MF, and WT = A then JA = A

R9: If MA = MF, and WT = L then JA = A

R10: If MA = AB, and WT = S then JA = W

R11: If MA = AB, and WT = A then JA = W

R12: If MA = AB, and WT = L then JA = W

R13: If MA = F, and WT = S then JA = RSAM

R14: If MA = F, and WT = A then JA = RASM

R15: If MA = F, and WT = L then JA = RASM

Table 2: The Fuzzy Rule Base Table

No	Device Availability (DA)	Job Waiting Time(JWT)	Job Allocation (JA)
1	One Free (OF)	Short (S)	Wait (W)
2	One Free (OF)	Average (A)	Wait (W)
3	One Free (OF)	Long (L)	Allocate (A)
4	Two Free (OF)	Short (S)	Wait (W)
5	Two Free (OF)	Average (A)	Allocate (A)
6	Two Free (OF)	Long (L)	Allocate (A)
7	Many Free (MF)	Short (S)	Allocate (A)
8	Many Free (MF)	Average (A)	Allocate (A)
9	Many Free (MF)	Long (L)	Allocate (A)
10	All Busy (AB)	Short (S)	Wait (W)
11	All Busy (AB)	Average (A)	Wait (W)
12	All Busy (AB)	Long (L)	Wait (W)
13	Failed (F)	Short (S)	Retrieve and Send to Available Machine(RSAM)
14	Failed (F)	Average (A)	Retrieve and Send to Available Machine(RSAM)
15	Failed (F)	Long (L)	Retrieve and Send to Available Machine(RSAM)

3.2.4 Defuzzification

The transformation from a fuzzy set to a crisp value is called defuzzification. The CoG is adopted in this study for defuzzification because its computational complexity is relatively high.

$$\mathrm{CoG}(B^0) = \frac{\sum_{q=1}^{Nq} \mu B0(yq)yq}{\sum_{q=1}^{Nq} \mu B0(yq)}$$

Where Nq is the number of quantization used to discretize membership function μB0(yq) of the fuzzy output B0. μB0(yq) is the degree of membership and yq are elements of the set.

Crisp Output = {Sum (Membership Degree * Singleton Position)}/(Membership degree). For instance with the output membership degree, are retrieving job and send to next available device = 0.0, wait = 0.5, allocate = 1.0 then the crisp value will be **Crisp Output** = (0.1*0.00)+ (0.5*0.50)+(1.0*1.00)/(0.0+0.5 +0.1) = 0.83 for this result therefore system allocates job.

3.3 Determining the job allocation

The job dispatcher performs fuzzy reasoning on the jobs, considering the job waiting time, the job due date and the job processing time in order to know the jobs to be dispatched to the individual parallel machines. The fuzzification module will receive the crisp numeric values as input, process them and map them into fuzzy membership function values. The fuzzy engine is responsible for processing all calculated membership function values using fuzzy sets' calculations and communicates with fuzzy rule base to identify the most suitable fuzzy output. However, the defuzzification module is responsible for converting the fuzzy output into a numeric output suitable for the environment decision and control situation. The fuzzy inference system determines the membership function for the job waiting time, the earliest due date and the job processing time which will be used in job allocation to the parallel machine.

3.3.1 Fuzzy Input/Output Specifications

The Fuzzy input/output variables are used for job dispatching and it is assumed to be the base for the dispatching rule. The fuzzy I/O specification is presented in Table 3. There are two

input variables and one output. The output is the effect of the result of input states.

3.3.2 Fuzzification.

The fuzzification is the process of transforming a crisp value into a fuzzy set, so that it can be used and processed by fuzzy inference mechanism. The inputs and output of the design as specified in Table 3 are assigned linguistic variables and some degrees of membership. For input one (the job due date) the corresponding range is close (0.0, 0.4), Distance (0.5, 1.0). For input two (the job waiting time) the corresponding range is short (0.0, 0.3), average (0.4, 0.6) and long (0.7, 1.0).

Tables 3. Fuzzy variables (Input/Output Specifications)

INPUT			**OUTPUT**
Due Date (DD)	**Waiting Time** (WT)	**Processing Time** (PT)	**Job Priority** (JP)
Close (C)	Short (S)	Very Short (VS)	Very Low (VL)
Distant(D)	Average (A)	Short (S)	Low (L)
	Long (L)	Medium (M)	Medium (M)
		Long (L)	High (H)
		Very Long (VL)	Very High (VH)

For input three (the job processing time), the corresponding range is very short (0.0, 0.1), short (0.2, 0.3), medium (0.4, 0.5), long (0.6, 0.8) and very long (0.9, 1.0). The levels for output (the job priority) are very low 0.0, low 0.25, medium 0.50, High 0.75 and very high 1.0.

3.3.3 Fuzzy Rules for job allocation

The rules are formulated using a series of if-then statements, combined with AND/OR operators. For instance, if the job due date is close AND the job waiting time is long AND the job processing time very short, then job should be very high. With three inputs, each having 10 membership functions, we have 2 * 3 * 5 = 30 rules as showed in table 3.2.

R1: If the job due date is close AND the job waiting time is Short AND the job processing time is very short THEN the job priority is high.

R3: If the job due date is close AND the job waiting time is long AND the job processing time is very short THEN the job priority is very high.

R7: If the job due date is close AND the job waiting time is Short AND the job processing time is very short THEN the job priority is high.

R11: If the job due date is close AND the job waiting time is Short AND the job processing time is medium THEN the job priority is high.

R17: If the job due date is distant AND the job waiting time is average AND the job processing time is very short THEN the job priority is medium.

R22: If the job due date is distant AND the job waiting time is Short AND the job processing time is medium THEN the job priority is low.

R27: If the job due date is distant AND the job waiting time is long AND the job processing time is long THEN the job priority is low.

R30: If the job due date is distant AND the job waiting time is long AND the job processing time is very long THEN the job priority is low.

Table 4: The Fuzzy Rule Base

No	**Due Date** (DD)	**Waiting Time** (WT)	**Processing Time** (PT)	**Job Priority** (JP)
1	Close (C)	Short (S)	Very Short (VS)	High (H)
2	Close (C)	Average (A)	Very Short (VS)	High (H)
3	Close (C)	Long (L)	Very Short (VS)	Very High (VH)
4	Close (C)	Short (S)	Short (S)	High (H)
5	Close (C)	Average (A)	Short (S)	High (H)
6	Close (C)	Long (L)	Short (S)	Very High (VH)
7	Close (C)	Short (S)	Medium (M)	High (H)
8	Close (C)	Average (A)	Medium (M)	High (H)
9	Close (C)	Long (L)	Medium (M)	High (H)
10	Close (C)	Short (S)	Long (L)	Medium (M)
11	Close (C)	Average (A)	Long (L)	Medium (M)
12	Close (C)	Long (L)	Long (L)	High (H)
13	Close (C)	Short (S)	Very Long (VL)	Medium (M)
14	Close (C)	Average (A)	Very Long (VL)	Medium (M)
15	Close (C)	Long (L)	Very Long (VL)	Medium (M)
16	Distance (D)	Short (S)	Very Short (VS)	Medium (M)
17	Distance (D)	Average (A)	Very Short (VS)	Medium (M)
18	Distance (D)	Long (L)	Very Short (VS)	High (H)
19	Distance (D)	Short (S)	Short (S)	Medium (M)
20	Distance (D)	Average (A)	Short (S)	Medium (M)
21	Distance (D)	Long (L)	Short (S)	Medium (M)
22	Distance (D)	Short (S)	Medium (M)	Low (L)
23	Distance (D)	Average (A)	Medium (M)	Low (L)

24	Distance (D)	Long (L)	Medium (M)	Medium (M)
25	Distance (D)	Short (S)	Long (L)	Very Low (VL)
26	Distance (D)	Average (A)	Long (L)	Low (L)
27	Distance (D)	Long (L)	Long (L)	Low (L)
28	Distance (D)	Short (S)	Very Long (VL)	Very Low (VL)
29	Distance (D)	Average (A)	Very Long (VL)	Low (L)
30	Distance (D)	Long (L)	Very Long (VL)	Low (L)

Using the values for each of the linguistic variable we have Very low = 0.0, low = 0.25, Medium =0.50, High =0.75, Very High = 1.00.

3.3.4 Defuzzification

The transformation from a fuzzy set to a crisp value is called defuzzification. The CoG is adopted in this study for defuzzification because its computational complexity is relatively high.

$$\mathrm{CoG}(B^0) = \frac{\sum_{q=1}^{Nq} \mu B0(yq)yq}{\sum_{q=1}^{Nq} \mu B0(yq)}$$

Where Nq is the number of quantization used to discretize membership function µB0(y) of the fuzzy output B0. µB0(y) is the degree of membership and yq are elements of the set.

Crisp Output = {Sum (Membership Degree * Singleton Position)}/(Membership degree). For instance, with the output membership degree, Very low = 0.0, low = 0.25, Medium =0.50, High =0.75, Very High = 1.00 then the crisp value will be **Crisp Output** = (0.1*0.00)+(0.3*0.25)+(0.9*0.50)+(0.6*0.75)+(0*1.00)/0.1+0.3 +0.9+0.6+0) = 0.51 for this result the job priority is medium.

4. EXPERIMENT AND RESULTS

Figure 2 shows machine failure handling in parallel machines for the case of one failed machine using the new job scheduling model. The figure shows the result of an experiment with four available machines which are meant to handle forty four loaded jobs. During the time of processing, it was noticed that one of the parallel machine (Hp LaserJet P2050 Series PCL6) that has jobs allocated to it went offline. It can be seen from the figure that the job allocated to the failed machine (Hp LaserJet P2050 Series PCL6) was reallocated to the three online machines (Hp LaserJet P2015 PCL6, Microsoft XPS Document Writer and Fax). In this way the system handles the issue of machine failure.

Figure 3 shows machine failure handling in parallel machines for the case of two failed machine using the new job scheduling model. The figure shows the result of an experiment with five available machines which are meant to handle forty four loaded jobs. During the time of processing, it was noticed that two of the parallel machines (Hp LaserJet P2050 Series PCL6 and Hp LaserJet P2200 Series PCL5) that has jobs allocated to them went offline. It can be seen from the figure that the job allocated to the failed machine (Hp LaserJet P2050 Series PCL6 and Hp LaserJet P2200 Series PCL5) was reallocated to the three online machines (Hp LaserJet P2015 PCL6, Microsoft XPS Document Writer and Fax). In this way the system handles the issue of machine failure.

Figure 4 shows machine failure handling in parallel machines for the case of four failed machine using the new job scheduling model. The figure shows the result of an experiment with seven available machines which are meant to handle forty four loaded jobs. During the time of processing, it was noticed that four of the parallel machines (Hp LaserJet P2050 Series PCL6, Hp LaserJet P2050 Series PCL6(Copy 1), Hp LaserJet P2300 Series PS and Hp LaserJet P2200 Series PCL5) that has jobs allocated to them went offline. It can be seen from the figure that the job allocated to the failed machine (Hp LaserJet P2050 Series PCL6, Hp LaserJet P2050 Series PCL6(Copy 1), Hp LaserJet P2300 Series PS and Hp LaserJet P2200 Series PCL5) was reallocated to the three online machines (Hp LaserJet P2015 PCL6, Microsoft XPS Document Writer and Fax). In this way the system handles the issue of machine failure.

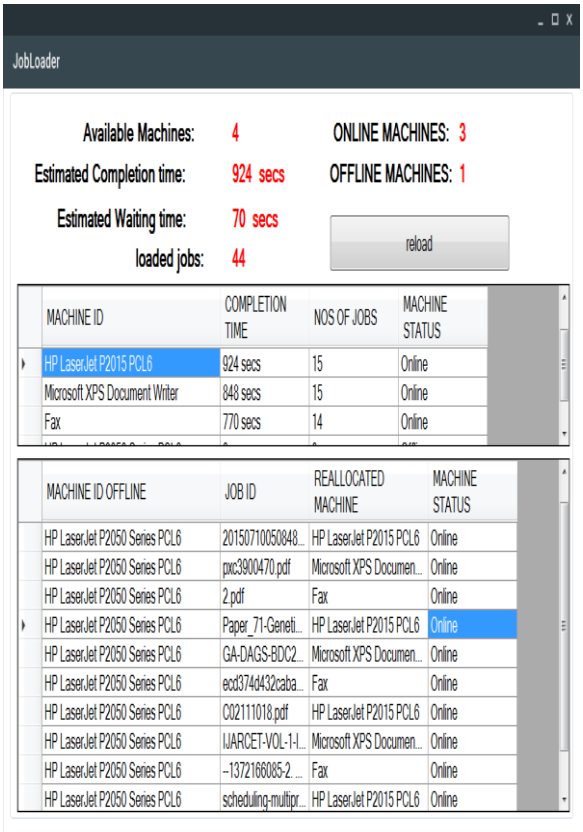

Figure 2: Handling machine failure in parallel machines (a case one machine failure) using the new job scheduling model.

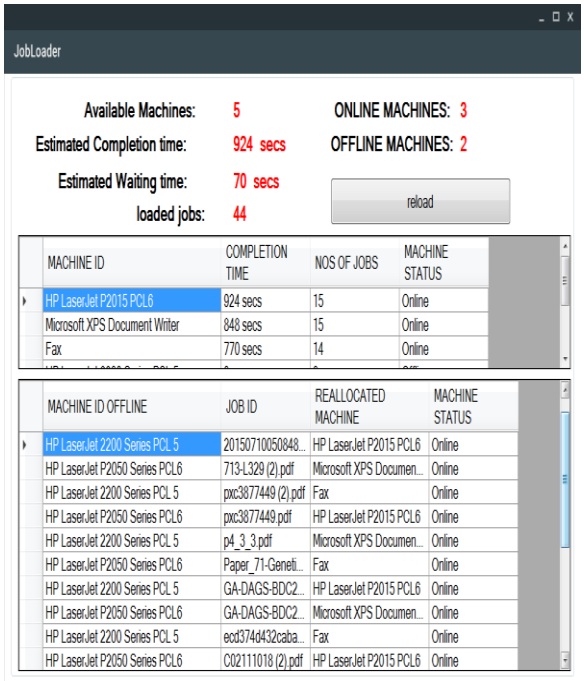

Figure 3: Handling machine failure in parallel machines (a case two machine failure) using the new job scheduling model.

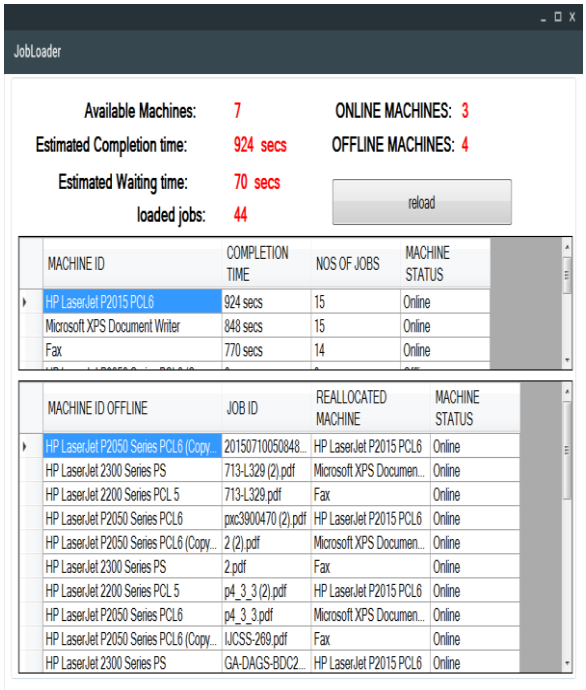

Figure 4: Handling machine failure in parallel machines (a case of four machine failures) using the new job scheduling model.

The new system was implemented to automatically generate the schedule for all the jobs uploaded in the system. This system generates values from the jobs uploaded to it and the value includes the job processing time and job waiting time. The generated jobs for testing this new model have the following characteristics:

a. Size of uploaded job ranges from 200 to 1200 jobs with an interval of 100 jobs.

b. For the new model only one job enters the system at a time.

c. The processing time for each job is a random number between 1 min and 4mins.

d. Number of machines used was fifteen for all the sizes of problems.

The developed model was used in the parallel machine environment. The result generated from the proposed system was compared with the result of other job scheduling algorithm like First Come First Serve (FCFS) scheduling model and genetic scheduling model. Three parameters are considered to evaluate the performance of the three different scheduling approaches. These parameters include the completion time, load balancing, and resource utilization.

The schedule length, average execution time, load balance and utilization can be computed from the generated result from the three tested scheduling models using the formula below:

$Schedule\ Length\ S_L = max\ \{E\_Time\ (\ )\}$

Machine Utilization

$$= \frac{\sum_{i=1}^{\text{no of machine}} \left(\frac{\text{Machine Execution time}}{\text{machine Scheduling length}}\right)}{\text{Total no of machines}}$$

Table 5: Analysis of the Machine Utilization from the Examined Job Scheduling Algorithms.

Number of jobs uploaded	Machine Utilization of the FCFS model	Machine Utilization of the GA model	Machine Utilization of the new model
200	83.46856%	89.57643%	98.2949%
300	81.87373%	90.79829%	98.6242%
400	85.45337%	90.76897%	99.0809%
500	84.57333%	91.50129%	99.1427%
600	88.09853%	94.90810%	99.2724%
700	89.92511%	95.63038%	99.4752%
800	90.91589%	95.80733%	99.4821%
900	91.68562%	95.65761%	98.9530%
1000	91.94647%	96.21440%	99.5082%
1100	93.33333%	97.23967%	99.4891%
1200	94.60927%	97.32404%	99.4525%

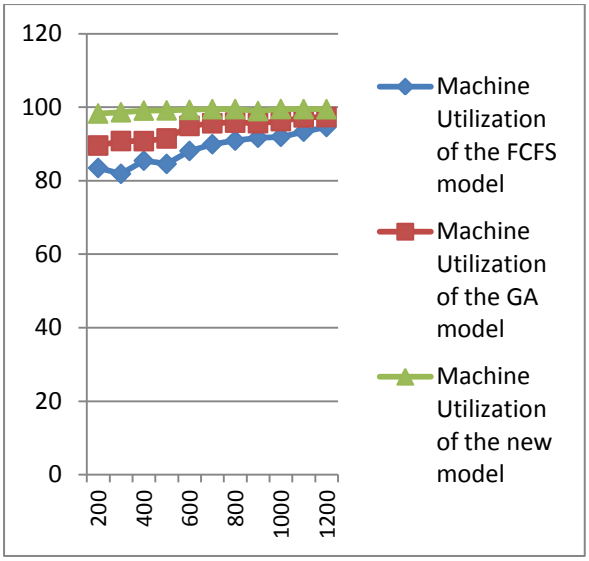

Figure 5: Graphical analysis of the machine utilization from the examined job scheduling algorithms.

Table 5: The analysis of the device utilization from the examined job scheduling algorithms. From the Table, we can observe the number of uploaded jobs alongside of the machine utilization for the FCFS model, the GA model, the new model. It can be seen from the result that when 300 jobs were uploaded the machine utilization for the new model, GA and FCFS models were 98.294%, 89.57643% and 83.46856% respectively. When 1300 jobs were uploaded the machine utilization for the new model, GA and FCFS models were 99.4525%, 97.32404% and 94.60927% respectively. This shows that the new model achieved high machine utilization when compared with other existing models.

Figure 5 shows graphical analysis of the machine utilization from the examined job scheduling algorithms. From the graph, it can be observed that the new model achieved better resource utilization that the FCFS model and the GA model; this shows that the new system achieved high machine utilization when compared with other existing models.

5. CONCLUSION

This paper has shown efficient job dispatching model that decreases the total time for jobs execution in identical parallel machine system. It is an effective job scheduling model that takes care of the different issues to obtain a good job schedule. The model was able to handle complexity issue in machine breakdown and some of these failures occurred even after jobs have been assigned to the individual machine. The system also considers the due date of the individual job, which is an advantage when compared with existing systems. The result generated from the experiment conducted in the course of this research has shown that the system achieves load balancing among the individual machine and high machine utilization, thereby minimizes the total job execution time. This paper has developed an enhanced model for job dispatch in identical parallel machines. This model has the capacity to handle the issue of machine breakdown, which is prevalent in the existing model during job execution in identical parallel machine.

6. REFERENCES

[1] Di-hua Sun, Wei He, Lin-Jiang Zheng and Xiao-yong Liao (2014), Scheduling flexible job shop problem subject to machine breakdown with game theory, 52 (13).

[2] Rachhpal S (2016), An Optimized Task Duplication Based Scheduling in Parallel System, *I.J. Intelligent Systems and Applications*, 8, 26-37

[3] Jasbir S and S. Gurvinder (2012), Task Scheduling using Performance Effective Genetic Algorithm for Parallel Heterogeneous System, *International Journal of Computer Science and Telecommunications 3(3); 233 – 245.*

[4] Weinberg J , (2002), "Job Scheduling on Parallel Systems", Job Scheduling Strategies for Parallel Processing, 5 (1), 67-73.

[5] Neelu S and S Sampada (2012), Task Scheduling Using Compact Genetic Algorithm for Heterogeneous System, *International Journal of Advanced Research in Computer Engineering & Technology, ISSN: 2278 – 1323, 1(3); 218- 221*

[6] Karthick K. U. (2011), "A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling", IJCSI International Journal of Computer Science Issues, 8(5), 1- 11, .

[7] Lei Z., C. Yuehui, S. Runyuan, J. Shan and Y. Bo, (2006) "A Task Scheduling Algorithm Based on PSO for Grid Computing", IEEE, 2 (1), 26-34.

[8] Abraham, R. B. and B. Nath. (2000), Nature's Heuristics for Scheduling Jobs on Computational Grids, The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), 5(3), 45-52,.

[9] Song S., Y. Kwok and K. Hwang, (2005). "Security-Driven Heuristics and A Fast Genetic Algorithm for Trusted Grid Job Scheduling", IEEE International Parallel and Distributed Processing, 4(2), 65-74.

[10] Orosz J.E and S. H. Jacobson. (2002) , Analysis of static simulated annealing algorithm, Journal of Optimization theory and Applications, 4(2), 165-182,.

[11] Braun R., H. Siegel., N. Beck, L. Boloni., M Maheswaran, A. Reuther, J. Robertson., M. Theys, B. Yao ,D Hensgen. and R. Freund.,(2001), "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing, 61: 810-837,

[12] Rajakumar S., V.P Arunachalam, and Selladurai V., (2006), Workflow balancing in parallel machine scheduling with precedence constraints using genetic algorithm, Journal of Manufacturing Technology Management, 17(2), 20-34.

[13] Safwat A. H and A. O Fatma (2016) Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment, *(IJACSA) International Journal of Advanced Computer Science and Applications, 7 (4). 112 – 122*

[14] Ramkumar R., A. Tamilarasi and T. Devi, (2011). Multi Criteria Job Shop Schedule Using Fuzzy Logic Control for Multiple Machines Multiple Jobs, International Journal of Computer Theory and Engineering, 3(2), 282-286

[15] Rachhpal S (2014),Task Scheduling in Parallel Systems using Genetic Algorithm, *International Journal of Computer Applications, 108(16); 0975 – 8887*

[16] Savas B (2012), Non-Identical Parallel Machine Scheduling With Fuzzy Processing Times Using Robust Genetic Algorithm And Simulation, *International Journal of Innovative Computing, Information and Control ICIC International ISSN 1349-4198, 8(1), 221 – 234*

[17] Mostafa R. M and H. A. A. Medhat (2011), Hybrid Algorithm for Multiprocessor Task Scheduling, *IJCSI International Journal of Computer Science Issues, 8(3), 14- 23,*

[18] Rachhpal S (2012),Task Scheduling With Genetic Approach and Task Duplication Technique, *International Journal of Computer Applications & Information Technology 1(1); 1-11*

[19] Kaleeswaran A, V Ramasamy and P. Vivekanandan, (2013). Dynamic Scheduling of Data Using Genetic Algorithm In Cloud Computing, *International Journal of Advances in Engineering & Technology,. ISSN: 2231-1963, 327, 5(2), 327-334.*

[20] Zubair K, S. Ravender and A. Jahangir, (2012), Tasks Allocation Using Fuzzy Inference In Parallel And Distributed System, *Journal Of Information And Operations Management, E-Issn: 0976-7762, 3(2); 322-326.*

[21] Ali M. A, (2012), A Fuzzy Dynamic Load Balancing Algorithm for Homogenous Distributed Systems, *International Journal of Computer, Electrical, Automation, Control and Information Engineering 6 (1); 1 - 11,*

[22] Mohammad S. G and E Mehdi (2013), High Performance Scheduling in Parallel Heterogeneous Multiprocessor Systems Using Evolutionary Algorithms, *I.J. Intelligent Systems and Applications.*2(1) 22 – 34

[23] Prabhjot K, and K. Amanpreet (2013), Implementation of Dynamic Level Scheduling Algorithm using Genetic Operators, *International Journal of Application or Innovation in Engineering & Management (IJAIEM), ISSN 2319 – 4847, 2 (7); 388 - 397.*

[24] Aparna V, V. Ramesh and U. P Sapna, (2014), Task Scheduling in Homogeneous Multiprocessor Systems Using Evolutionary Techniques, *International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459,4(2);77 – 86*

[25] Leila A, (2014), Solving The Job Shop Scheduling Problem With A Parallel And Agent-Based Local Search Genetic Algorithm, *Journal Of Theoretical And Applied Information Technology, Issn: 1992-8645, 62(.2); 1958-1969*

[26] Selvi. V (2014), Multi Objective Optimization Problems On Identical Parallel Machine Scheduling Using Genetic Algorithms, International Journal on Recent Researches in Science, Engineering & Technology, 2 (7) 112 - 122,

[27] Zeinab K. and J. M. Seyed, (2014), A Novel Decentralized Fuzzy Based Approach for Grid Job, Journal of Telecommunication, Electronic and Computer Engineering, , ISSN: 2180 - 1843 6 (1); 1-12

[28] Nirmala H and H A Girijamma (2014), Fuzzy Scheduling Algorithm for Real –Time multiprocessor system, International Journal of Scientific & Engineering Research, 5(7); 2229-5518.

[29] Seyed M. H,and H. T. Sai, (2015), A Fuzzy Genetic Algorithm for Scheduling of Handling/Storage Equipment in Automated Container Terminals, *IACSIT International Journal of Engineering and Technology, 7(6); 234- 245*