SCHEDULING GANGS IN A DISTRIBUTED SYSTEM

HELEN D. KARATZA

Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, GREECE
Email: karatza@csd.auth.gr

Abstract: In this paper we study the performance of parallel job scheduling in a distributed system. A special type of scheduling called *gang scheduling* is considered. In gang scheduling jobs consist of a number of interacting tasks, which are scheduled to run simultaneously on distinct processors. Two gang scheduling policies are used to schedule parallel jobs for two different types of job parallelism. Also, we present the average performance of all jobs as well as the relative performance of small and large gangs. We examine various workloads using simulation techniques. The results show that the policy, which gives priority to large gangs performs better than the one that does not take job characteristics into account while making the ordering decision.

Keywords: Simulation, Performance, Distributed Systems and Gang Scheduling.

1 INTRODUCTION

Distributed systems have drawn considerable attention over many years. They consist of several, loosely interconnected processors, where jobs to be processed are in some way apportioned among the processors and various techniques are used to coordinate processing. However, it is not clear to how efficiently schedule parallel jobs. To determine this, it is critical to properly assign the tasks to processors and then schedule execution on distributed processors. Good scheduling policies can maximize system and individual application performance and avoid unnecessary delays.

In this study jobs consist of parallel tasks that are scheduled to execute concurrently on a set of processors. The parallel tasks need to start essentially at the same time, co-ordinate their executions and compute at the same pace. This type of resource management is called "gang scheduling" or "co-scheduling". It allows tasks to interact efficiently by using busy waiting, without the risk of waiting for a task that is not currently running.

Because gang scheduling demands that no task execute unless all other gang member tasks execute, some processors may remain idle even when there are tasks waiting to be run. With gang scheduling, at any time there is a one-to-one mapping between tasks and processors. Although the total number of tasks in the system may be larger than the number of processors, no gang contains more tasks than it does processors. We assume that all the tasks within the same gang execute for the same amount of time, i.e. the computational load is balanced between them.

Gang scheduling in distributed systems and multiprogrammed parallel systems has been studied by many authors, such as [Aida, 2000], [Feitelson and Rudolph, 1996], [Feitelson and Jette, 1997], [Frachtenberg et al, 2005], [Karatza, 1999a], [Karatza, 1999b], [Karatza, 2001a], [Karatza, 2001b], [Karatza, 2002], [Karatza, 2003], [Karatza and Hilzer, 2004], [Sobalvarro and Weihl, 1995], [Squillante et al, 1996], [Wang et al, 1997], [Wiseman and Feitelson, 2003], [Zhang et al, 2003a] and [Zhang et al, 2003b].

Only distributed systems are considered in this paper. Simulation models are used to answer performance questions about the performance of scheduling policies in cases of different job parallelism. The design choices considered include different ways to schedule gangs. We compare the performance of two known gang-scheduling policies for different workload models each of which has certain characteristics related to the number of processors requested by a job and to the system load.

Mechanisms how job size characteristics affect job scheduling performance also are investigated in [Aida, 2000 and Karatza 2001b]. However, in these papers a parallel system with a single waiting queue is studied, while in our study we consider a distributed system where each processor is equipped with its own queue. Furthermore, we address issues related to the performance of different job classes (small and large gangs) and we consider fairness of job classes. To our knowledge, the analysis of gang scheduling in queueing network models of distributed systems operated under our workload models does not appear elsewhere in the research literature.

The structure of this paper is as follows. Section 2.1 introduces system and workload models, section 2.2 describes the scheduling policies and section 2.3 presents the metrics used to assess the performance of the scheduling policies. The model implementation and its input parameters are described in section 3.1, while the simulation results are both presented and analysed in section 3.2. Finally, section 4 summarizes the paper and provides recommendations for further research.

2 MODEL AND METHODOLOGY

2.1 System and Workload Models

In our model we consider an open queuing network model consisting of $P = 32$ distributed homogeneous and independent processors each served by its own queue (Figure 1). They are interconnected by a high-speed network with negligible communication delays. Processors P_{x1}, P_{x2}, .. P_{xy} are allocated to a parallel job x, which has y parallel tasks.

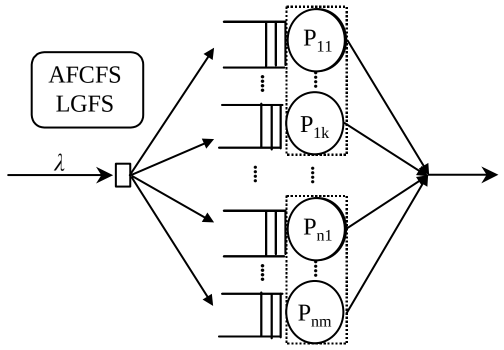

Figure 1. The queuing network model, where λ is the mean job inter-arrival time.

Jobs are *gangs* i.e. they consist of tasks, which execute concurrently on processor partitions, where each task starts at the same time and computes at the same pace. At any time, there is a one-to-one mapping between tasks and processors. We assume that all tasks within the same gang execute for the same amount of time. Each job begins execution only when a sufficient number of idle processors are available to meet its needs.

We consider that every job x consists of $t(x)$ tasks where $1 \le t(x) \le P/j$. Two cases for j are considered, $j = 1, 2$. Therefore, in one case we bound the number of tasks per job by 32, while in the other case the number of tasks per job is bounded by 16.

The number of tasks in a job is the job's *degree of parallelism*. If $p(x)$ represents the number of processors required by job x, then the following relationship holds:

$$1 \le t(x) = p(x) \le P/j$$

The number of tasks in job (gang) x is called the "size" of job x. We call a job "small" ("large") if it requires a small (large) number of processors. In our model, jobs that consist of y tasks where $1 \le y \le 4$ are characterized as *small*, while they are characterized as *large* when $5 \le y \le P/j$, for $j = 1, 2$.

Routing policy of gang tasks. Processor queues are sorted into decreasing queue length order and then a variation of the "join the shortest queue" policy is applied. That is, the $t(x)$ tasks that belong to a gang x are assigned to the shortest $t(x)$ of the P queues, every task to a different processor queue.

Tasks in processor queues are examined in the order according to the scheduling policy. A job x starts to execute only if all $p(x)$ processors assigned to it are available. Otherwise, all job x tasks wait in their assigned queues. When a job finishes execution, all processors assigned to it are released. The number of jobs that can be processed in parallel depends on the following: (i) Job size and (ii) Scheduling policy that is employed.

The technique used to evaluate the performance of the scheduling disciplines is experimentation using a synthetic workload simulation.

The workload considered here is characterized by three parameters: (i) The distribution of job inter-arrival time, (ii) The distribution of gang sizes and (iii) The distribution of task service demand.

We assume that there is no correlation between the different parameters. For example, a gang with a small number of tasks may have a long execution time.

2.1.1 Distribution of job inter-arrival times

We consider that job inter-arrival times are exponential random variables with a mean of $1/\lambda$.

2.1.2 Distribution of gang size

We assume that the number of tasks of jobs is uniformly distributed in the range of [1..P/j], where j = 1, 2. Therefore, the mean number of tasks per job is equal to the $\eta = (1+P/j)/2$.

2.1.3 Service time distribution

Service demands of gang tasks are exponentially distributed with a mean of $1/\mu$.

2.2 Scheduling Strategies

We assume that the scheduler has perfect information when making decisions, i.e. it knows the exact number of processors required by each job. We now describe the scheduling strategies employed. We assume that the scheduling overhead is negligible.

Adapted First-Come-First-Served (***AFCFS***). This method attempts to schedule a job whenever processors assigned to its tasks are available. When there are not enough processors available for a large job whose tasks are waiting in the front of the queues, AFCFS policy schedules smaller jobs whose tasks are behind the tasks of the large job. One major problem with this scheduling policy is that it tends to favor those jobs requesting a smaller number of processors at the expense of larger jobs. Thus the response time of large gangs increases.

Largest-Gang-First-Served (***LGFS)***. With this policy tasks are placed in increasing job size order in processor queues (tasks that belong to larger gangs are placed at the head of queues). All tasks in the queues are searched in relative order and the first jobs whose assigned processors are available only then begin execution. This method tends to improve the performance of large, highly parallel gangs at the expense of smaller gangs, but in many computing environments this discrimination is acceptable, if not desirable. For example, supercomputer centers often run large, highly parallel jobs that cannot run elsewhere.

2.3 Performance Metrics

Response time of a random job (gang) is the interval of time from the dispatching of this job tasks to processor queues to service completion of this job (time spent in processor queues plus time spent in service). Parameters used in later simulation computations are presented in Table 1.

Table 1. Notations

P	number of distributed processors
λ	mean job arrival rate
μ	mean processor service rate
U	mean processor utilization
RT	mean response time of all gangs
RT_s	mean response time of small gangs
RT_l	mean response time of large gangs
MRT_s	maximum RT_s
MRT_l	maximum RT_l

Overall job performance is determined by RT. Small and large gang performance is determined by RT_s and RT_l respectively. Maximum RT_s and maximum RT_l are used as an indication of fairness in small and large gangs service respectively. Internal efficiency is primarily represented by mean processor utilization because it indicates the level of contention for the most critical system resources.

3 SIMULATION RESULTS AND DISCUSSION

3.1 Model Implementation and Input Parameters

The queuing network model is simulated with discrete event simulation modeling [Law and Kelton, 1991] using the independent replication method. For each set of workload parameters we run 30 replications of the simulation with different seeds of random numbers and for 32,000 served jobs in each replication. For every mean value, a 95% confidence interval is evaluated. All confidence intervals are less than 5% of the mean values.

In the simulation experiments we defined mean processor service time: $1/\mu = 1$.

With regard to mean inter-arrival time of jobs we considered the following two cases:

(a) Gang size varies in the range of 1..32. We chose:

$$1/\lambda = 0.73,\ 0.74,\ 0.75,\ 0.76.$$

The value $1/\lambda = 0.73$ is chosen as starting point for the experiments because the processors average 16.5 tasks per job. When all processors are busy, an average of 1.94 jobs can be served in each unit of time. This implies that the arrival rate has to be less than 1.94. So, we had to choose a λ such that it hold the conditions $1/\lambda > 1 / 1.94 = 0.516$, i.e. the processors queues will not be saturated. However, due to gang scheduling there are often idle processors although there are tasks in the respective queues. Therefore the queues get very easily saturated when mean inter-arrival time is close to 0.516. After experimental runs with various values of $1/\lambda$ we chose 0.73 as a the smallest mean inter-arrival time for the experiments.

(b) Gang size varies in the range of 1..16. In this case processors average 8.5 tasks per job. We chose:

$$1/\lambda = 0.376,\ 0.381,\ 0.386,\ 0.392.$$

The reason for choosing these values for mean-inter-arrival time in the case b) is because in a system where arriving jobs consist of parallel tasks with mean number of tasks per job η, the expected mean processor utilization is:

$$U = \frac{\lambda \cdot \eta}{P \cdot \mu}.$$

Now, based on this formula, in order to compare the performance of the two scheduling policies under similar system load in the two different cases of gang size distribution, we must consider:

$$\lambda_a * 16.5 = \lambda_b * 8.5,$$

where λ_a and λ_b are mean arrival rates in cases a) and b) respectively. However in this paper parallel jobs are gangs. Due to different fitting of gangs to available processors in the 1..32 and 1..16 gang size cases we expect that the mean processor utilization will not be exactly the same in the corresponding λ_a and λ_b cases. However, it can still be considered as comparable.

It should be noted that in this study gangs consisting of 1-4 tasks are characterized small. Therefore, small gangs are a smaller part of the total number of jobs z in the 1..32 case (z/8) than in the case of 1..16 (z/4).

3.2 Performance Analysis

Tables 2 and 3 show mean processor utilization in the 1..32 and 1..16 gang size distribution cases respectively.

Table 2. U versus $1/\lambda$
(1..32 gang size distribution case)

$1/\lambda$	AFCFS	LGFS
0.76	0.675	0.677
0.75	0.683	0.685
0.74	0.690	0.695
0.73	0.696	0.704

Table 3. U versus $1/\lambda$
(1..16 gang size distribution case)

$1/\lambda$	AFCFS	LGFS
0.392	0.662	0.670
0.386	0.668	0.679
0.381	0.673	0.687
0.376	0.681	0.694

The relative difference in performance of AFCFS and LGFS policies is depicted in Figures 2 and 7 (with regard to overall gangs performance) and also in Figures 3-6 and 8-11 (with regard to relative performance of small and large gangs).

Figures 2-6 correspond to the 1..32 case and Figures 7-11 correspond to the 1..16 case. Figures 2 and 7 show the ratio of RT versus $1/\lambda$ when LGFS is compared to AFCFS. Figures 3 and 8 show the ratio RT_l / RT_s in the AFCFS and LGFS cases. Figures 4 and 9 present the ratio MRT_l / MRT_s in the AFCFS and LGFS cases. Figures 5 and 10 depict the ratio of RT_s and the ratio of RT_l versus $1/\lambda$ when LGFS is compared to AFCFS. Figures 6 and 11 present the ratio of MRT_s and the ratio of MRT_l versus $1/\lambda$ when LGFS is compared to AFCFS.

The results demonstrate that in both 1..32 and 1..16 cases, utilization is slightly larger when using LGFS (Tables 2-3) because LGFS schedules jobs on the available processors more efficiently than AFCFS method does.

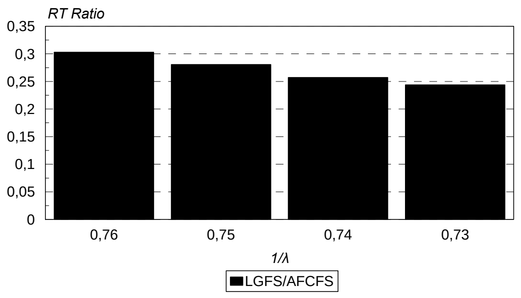

Figure 2. *RT Ratio* versus $1/\lambda$
(1..32 gang size distribution case)

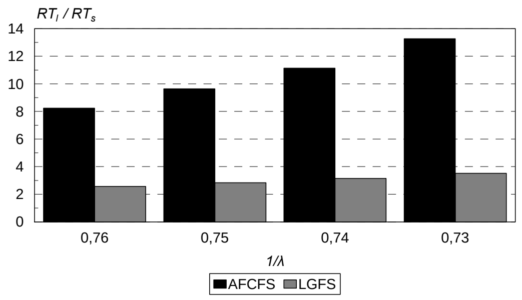

Figure 3. RT_l / RT_s *Ratios* versus $1/\lambda$
(1..32 gang size distribution case)

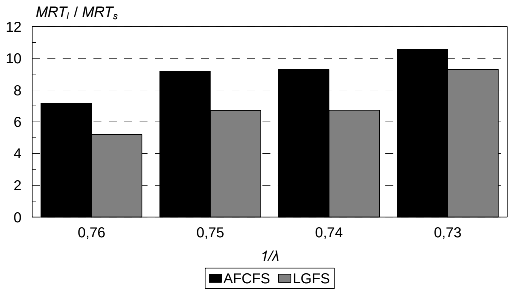

Figure 4. MRT_l / MRT_s *Ratios* versus $1/\lambda$
(1..32 gang size distribution case)

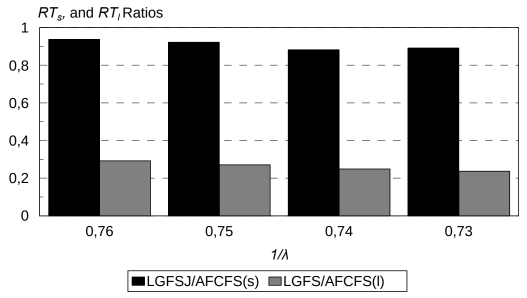

Figure 5. RT_s and RT_l *Ratios* versus 1/λ (1..32 gang size distribution case)

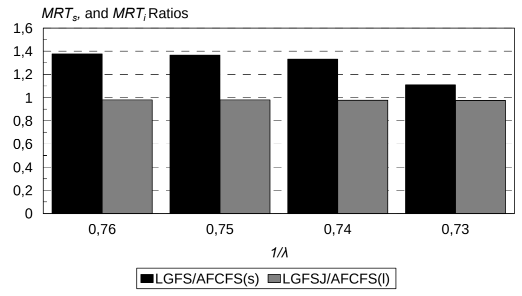

Figure 6. MRT_s and MRT_l *Ratios* versus 1/λ (1..32 gang size distribution case)

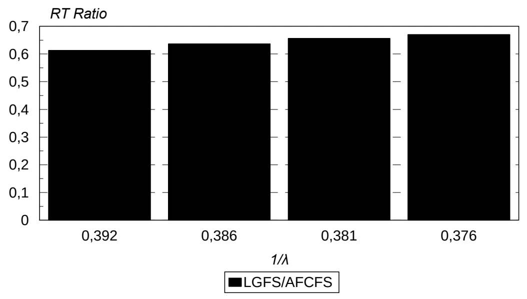

Figure 7. *RT Ratio* versus 1/λ (1..16 gang size distribution case)

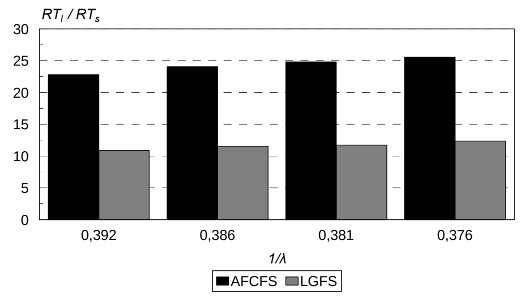

Figure 8. RT_l / RT_s *Ratios* versus 1/λ (1..16 gang size distribution case)

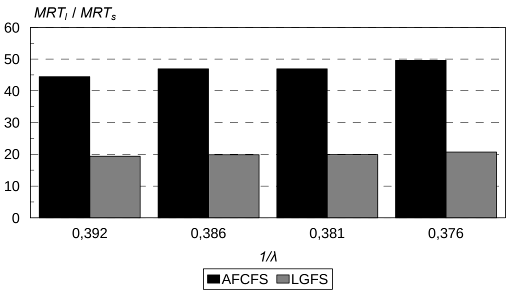

Figure 9. MRT_l / MRT_s *Ratios* versus 1/λ (1..16 gang size distribution case)

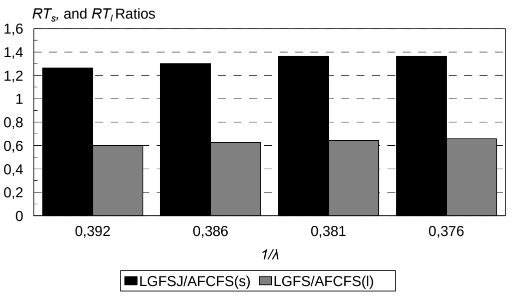

Figure 10. RT_s and RT_l *Ratios* versus 1/λ (1..16 gang size distribution case)

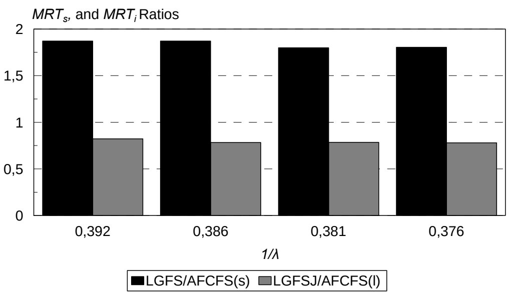

Figure 11. MRT_s and MRT_l *Ratios* versus 1/λ (1..16 gang size distribution case)

Regarding the mean response time of all jobs, in all cases the LGFS method performs better than AFCFS. This is shown in Figures 2 and 7, where *RT* is lower with the LGFS scheduling strategy than with the AFCFS. Therefore, from the two methods that we examine, LGFS is the best method as far as it concerns overall performance. This is in accordance to previous research conclusions [Karatza, 1999a]) where the method that gives priority to larger gangs performs better than the AFCFS method. However, in [Karatza, 1999a] a closed queueing network model is studied with a fixed number of jobs, whereas in this paper the queuing network model is open.

In Figure 2 we observe that in the 1..32 case the *RT* ratio slightly decreases with increasing load. Therefore the superiority of the LGFS over AFCFS increases with increasing load. This is because it is more probable at high loads than at low loads large gangs to be blocked when the AFCS policy is employed. Therefore the advantages of the LGFS method in the 1..32 case are better exploited at high loads.

In Figure 7 it is evident that in the 1..16 case the response times in the two gang scheduling policies cases differ to smaller degree than in the 1-32 case. This is because the variability in gangs size is smaller in this case than in the case of 1..32. The large gangs in the 1..16 case have a maximum size of 16 tasks. Therefore, when the AFCFS method is used, large gangs in this gang size distribution case can be scheduled on idle processors easier than the large gangs of the 1..32 case. This is the reason that the superiority of the LGFS method over AFCFS is less significant in the 1..16 case than in the case of 1..32.

In Figure 7 it is also evident that in the 1..16 case *RT* ratio slightly increases with increasing load. Therefore the two scheduling methods tend to perform slightly closer with increasing load. Further to the comments of the previous paragraph, this is because, larger the load, larger is the possibility for the AFCFS method to find a larger number of suitable jobs to fit on the available processors.

In Figures 3 and 8 it is shown that in all cases large gangs have larger mean response time than small gangs. This is because small gangs can easier find enough available processors to serve them than large gangs can. Comparing the RT_l / RT_s ratio in the two scheduling policies cases we observe that the smallest difference in performance between large and small gangs is in the LGFS case. This is because with the LGFS method large gangs are given priority over small gangs. Therefore the difference in the mean response time of large and small gangs is smaller in the LGFS case than in the case of AFCFS.

Also, in Figures 3 and 8 it is shown that in all cases the difference in performance between the two job classes increases with increasing load. This is due to the fact that there are fewer small gangs to overcome large gangs at light loads than at heavy loads. Therefore, the LGFS scheduling policy provides an advantage for fairer service to large gangs as compared to the situations where AFCFS is used. Thus, LGFS can be exploited better for high loads than for low ones. The RT_l / RT_s ratios are smaller in the 1..32 case than in the case of 1..16.

The results shown in Figures 4 and 9 demonstrate that observations similar to those that hold for the RT_l / RT_s ratios also hold for the ratios MRT_l / MRT_s. That is, the maximum response time of large gangs is larger than the maximum response time of small gangs. Furthermore, the difference in the two gang classes maximum response times is smaller in the LGFS case. The RT_l / RT_s ratios increase with increasing load and they are larger in the 1..16 case than in the case of 1..32.

As it was expected, the results in Figures 5 and 10 show that the mean response time of large gangs in the LGFS case is lower than in the AFCFS case. Furthermore, in the 1..16 case the small gangs perform worst in the LGFS case than in the AFCFS case. However, in the 1..32 case the short gangs perform better in the LGFS case than in the case of AFCFS.

There is the following explanation why small gangs although they are given lower priority than large gangs with the LGFS method, in the 1..32 case they have shorter mean response time with this method than with the AFCFS: The variability in gang size is larger in the 1..32 case than in the case of 1..16. In the former case, when a very large gang fits in the available processors, then the remaining processors can fit easier to small gangs than to medium sized gangs. However, in the 1..16 case, the large gangs have a maximum length of 16. Therefore, when the LGFS method is used, it is possible for medium sized gangs to fit in the available processors and consequently small gangs have to delay in their queues.

It should be noted though that in both cases of gang size distribution the extent of large gang performance improvement with the LGFS policy is much more significant than the performance improvement (1..32 case) or the performance deterioration (1..16 case) of small gangs. For this reason there is an overall performance improvement with the LGFS policy.

In Figures 6 and 11 it is shown that small gangs have higher *MRT* in the LGFS case than in the case of AFCFS. In the 1..16 case *MRT* of large gangs is smaller in the LGFS case than in the case of AFCFS. However, in the 1..32 case the *MRT* of large gangs does not differ significantly in the two scheduling policies cases. This is due to the fact that in this gang size distribution case, large gangs have a maximum size of 32, which is equal to the number of processors. Therefore, when a gang of a very large size arrives to the system when the processors are serving at least two other gangs, it is possible for this gang to experience long delay in the queues with either scheduling method.

4 CONCLUSIONS AND FURTHER RESEARCH

This paper examines the performance of two gang scheduling policies in a distributed system. The average performance of all gangs as well as the relative performance of small and large gangs are studied. Two different cases of job parallelism are examined. Various workload conditions are considered using simulation techniques.

The objective in this paper is to identify conditions that produce good overall performance in terms of mean response time of all jobs while preserving fairness of small and large gangs service.

The simulation results reveal that from the two scheduling methods that we examine the LGFS method provides the best overall performance. Furthermore, LGFS provides fairer service to individual job classes. This is achieved by preventing small gangs to arbitrarily overtake large gangs. Thus, the difference in performance of the two gang classes is smaller in the LGFS case than in the case of AFCFS. Also the results show that the overall performance and the relative performance of the two gang classes depend on the workload.

As a future research we plan to examine cases where along with gangs there are also parallel jobs, which consist of independent tasks, which can execute on any processor and in any order. Also, we plan to examine gang task service demands with large variability.

REFERENCES

Aida K. 2000, “Effect of Job Size Characteristics on Job Scheduling Performance”. In *Job Scheduling Strategies for Parallel Processing*, *Lecture Notes in Computer Science*, Springer-Verlang, Berlin, Germany. Vol. 1911. Pp1-10.

Feitelson D.G. and Rudolph L. 1996, “Evaluation of Design Choices for Gang Scheduling Using Distributed Hierarchical Control”. *Journal of Parallel and Distributed Computing*, Academic Press, New York, USA. Vol. 35. Pp18-34.

Feitelson D.G. and Jette M.A. 1997, “Improved Utilisation and Responsiveness with Gang Scheduling”. In *Job Scheduling Strategies for Parallel Processing*, *Lecture Notes in Computer Science*, Springer-Verlang, Berlin, Germany. Vol. 1291. Pp238-261.

Frachtenberg E., Feitelson D.G., Petrini F. and Fernandez J. 2005, “Adaptive Parallel Job Scheduling with Flexible Coscheduling”. *IEEE Transactions on Parallel and Distributed Systems*, IEEE Computer Society, Los Alamitos, CA, USA. Vol. 16(11). Pp1066-1077.

Karatza H.D. 1999a, “A Simulation-Based Performance Analysis of Gang Scheduling in a Distributed System”. In *Proc. of the 32nd Annual Simulation Symp.* (San Diego, CA, USA, April) IEEE Computer Society, Los Alamitos, CA, USA. Pp26-33.

Karatza, H.D. 1999b, “Gang Scheduling in a Distributed System with Processor Failures”. In *Proc. of the UK Performance Engineering Workshop* (Bristol, UK, July) University of Bristol, Bristol, UK. Pp199-208.

Karatza H.D. 2001a, “Performance Analysis of Gang Scheduling in a Distributed System Under Processor Failures”. *International Journal of Simulation: Systems, Science & Technology*, UK Simulation Society, Nottingham, UK. Vol. 2(1). Pp14-23.

Karatza H.D. 2001b, “Gang Scheduling Performance under Different Distributions of Gang Size”. *Parallel and Distributed Computing Practices*, Nova Science Publishers, Hauppauge, NY, USA. Vol. 4(4). Pp433-449.

Karatza H.D. 2002, “Gang Scheduling Performance on a Cluster of Non-Dedicated Workstations”. In *Proc. of the 35th Annual Simulation Symposium* (San Diego, California, April) IEEE Computer Society, Los Alamitos, CA, USA. Pp115-121.

Karatza H.D. 2003, “Gang Scheduling in a Distributed System under Processor Failures and Time-varying Gang Size”. In *Proc. of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems* (San Juan, Puerto Rico, May) IEEE Computer Society, Los Alamitos, CA, USA. Pp330-336.

Karatza H.D. and Hilzer R.C. 2004, “Scheduling Sequential Jobs and Gangs in a Distributed Server System”. In *Proc. of the 5th EUROSIM Congress on Modelling and Simulation (Special Session on Modelling and Simulation of Distributed Systems and Networks),* (SCité Descartes, Marne la Vallée, France, September) EUROSIM–FRANCOSIM–ARGESIM, Paris, France. Pp17-22 (CD).

Law A. and Kelton D. 1991, *Simulation Modeling and Analysis*. 2nd Ed., McGraw-Hill, Inc, New York, USA.

Sobalvarro P.G. and Weihl W.E. 1995, “Demand-based Coscheduling of Parallel Jobs on Multiprogrammed Multiprocessors”. In *Job Scheduling Strategies for Parallel Processing*, *Lecture Notes in Computer Science*, Springer-Verlang, Berlin, Germany. Vol. 949. Pp106-126.

Squillante M.S., Wang F. and Papaefthymioy M. 1996, “Stochastic Analysis of Gang Scheduling in Parallel and Distributed Systems”. *Performance*

Evaluation, Elsevier, Amsterdam, Holland. Vol. 27&28 (4). Pp273-296.

Wang F., Papaefthymiou M. and Squillante M.S. 1997, “Performance Evaluation of Gang Scheduling for Parallel and Distributed Systems”. In *Job Scheduling Strategies for Parallel Processing*, *Lecture Notes in Computer Science*, Springer-Verlang, Berlin, Germany. Vol. 1291. Pp184-195.

Wiseman Y. and Feitelson D.G. 2003, “Paired Gang Scheduling”. *IEEE Transactions on Parallel and Distributed Systems*, IEEE Computer Society, Los Alamitos, CA, USA. Vol. 14(6). Pp581-592.

Zhang Y., Franke H., Moreira J. and Sivasubramaniam A. 2003a, “An Integrated Approach to Parallel Scheduling Using Gang-Scheduling, Backfilling and Migration”. *IEEE Transactions on Parallel and Distributed Systems*, IEEE Computer Society, Los Alamitos, CA, USA. Vol. 14(3). Pp236-247.

Zhang Y., Yang A., Sivasubramaniam A. and Moreira J. 2003b, “Gang Scheduling Extensions for I/O Intensive Workloads”. In *Job Scheduling Strategies for Parallel Processing*, *Lecture Notes in Computer Science*, Springer-Verlang, Berlin, Heidelberg, Germany. Vol. 2862. Pp183-207.

BIOGRAPHY

HELEN D. KARATZA is an Associate Professor in the Department of Informatics at the Aristotle University of Thessaloniki, Greece. Her research interests mainly include Performance Evaluation of Parallel and Distributed Systems, Multiprocessor Scheduling, Cluster Computing and the

Grid, Mobile Agents, Mobile Computing and Simulation. Dr. Karatza is a member of the Editorial Board of the International Journal of Simulation: Systems, Science & Technology (the UK Simulation Society). She has served as a member of Program Committees and Program Chair of several Simulation related International Conferences/Symposia. Her email and web address are <karatza@csd.auth.gr> and <agent.csd.auth.gr/~karatza>.