Parallel Crew Scheduling on Workstation Networks Using PVM*

Christos Goumopoulos[1], Efthymios Housos[1] and Olle Liljenzin[2]

[1] Department of Electrical & Computer Engineering, University of Patras, GR-26500, Greece
[2] Carmen Systems AB, S-41103 Gothenburg, Sweden
Tel.: +30 61 997304, Fax: +30 61 997316
E-mail:{goumop, housos}@ee.upatras.gr, olle@carmen.se

Abstract. In this paper the ability to efficiently solve large crew scheduling problems on a network of workstations (NOW) is presented. Large crew scheduling problems from the Lufthansa set of problems have been solved with a near linear speedup on the generator component of the problem. The generator is the most time consuming component of the solution process, which implies that a significant improvement of the overall solution process is possible. This paper presents the first tangible results of the HPCN Esprit project PAROS, where the complete crew scheduling procedure on a NOW is parallelized and extended.

1 Introduction and Motivation

The long term planning and the scheduling of resources in the transportation industry is a very complex and time consuming process. This process can be usually seen as a sequence of phases which proceed sequentially and have minor feedback links. The following figure illustrates the typical planning process in the airline industry.

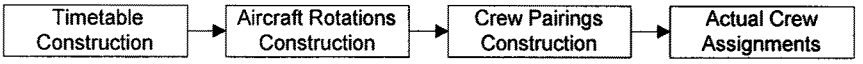

Fig. 1. The resource planning process in the airline industry

The main problem every airline must solve, after the construction of its aircraft rotations for a given planning horizon, involves the optimal construction of crew pairings and their assignment to its crew members [1]. Given a set of flights (*legs*) that an airline must fly, the crew scheduling problem is to find a set of legal round-trips (*pairings*) to satisfy all the contractual and safety rules of the airline while covering each flight in the timetable at a minimum cost. The crew pairings can only start and end at pre-defined crew bases. The solution of the crew scheduling problem for a large airline is a very time consuming and deadline driven procedure. All airlines would prefer to schedule their crews as late as possible, in order to minimize possible fleet changes and also give the marketing department time to better gage the market needs.

In recent years, many of the European airline companies have invested in automatic tools for crew planning and scheduling. Carmen Systems has developed an automatic crew scheduling product [2], that is currently in production use by many of the major European airlines. However, the runtime required for the solution of large

* This work is partly funded by the European Commission, within the HPCN domain of the Esprit programme (project PAROS, No. 20.115)

fleets is quite long and it often requires expert planner assistance. For example, at present, a typical problem involving the scheduling of a medium size fleet at Lufthansa requires 10-15 hours of computing, while for large problems, such as the monthly schedule, as much as 150 hours are needed [7].

The availability of fast workstations has allowed the airline companies to reduce the use of mainframes and thus significantly reduced their operational costs. However, the proliferation of workstations is in part responsible for the reduced efficiency of the hardware usage. This is in part resolved with the use of faster networks, and more efficient message passing systems which allow for the full exploitation of idle CPU cycles and other global resources. These networks of workstations (NOW) can be used to execute in parallel most of the time consuming components of the crew scheduling process. This is the precise problem that the PAROS Esprit project is also attempting to address.

In the context of the PAROS (Parallel Large Scale Automatic Scheduling) project the consortium is faced with the challenge of improving the performance and extending the functionality of the Carmen automatic crew scheduling process on a NOW. Lufthansa is the co-ordinating partner of the project and has an immediate need of the project deliverables. Lufthansa also possesses a large number of powerful workstations connected with a fast network. The results from the first project prototypes demonstrate that it is possible to significantly improve the speed of the crew scheduling process and extend the practical range of solvable problems.

The remaining of the paper is organized as follows. In section 2 we introduce briefly the Carmen system, in section 3 we present the parallelization approach and give a high level description of some implementation issues. Experimental results and performance issues are shown in section 4, followed by our conclusions in section 5.

2 The Carmen System

The solution process used to solve the crew scheduling problem in most successful production systems involves two major steps: *the generation of a large set of pairings* and *the selection of the best solution pairings (set covering optimization)*. After the generation of a large set of pairings, a very small subset of these pairings is selected such that every flight leg is covered by a pairing and the total solution cost is minimized. The number of legs covered in every run is from 100 to 7,000 and as many as 1,000,000 pairings are often generated.

An overview of the automatic crew scheduling solution process in the Carmen system can be seen in Figure 2. In the beginning an initial solution is created using various heuristics, and then subproblems of reasonable size are sequentially selected for optimization [3]. In the pairing generator, basically consisting of a depth-first search algorithm, the search tree is pruned based on the user controlling parameters and a complex rule system in order to avoid excessive generation of useless or illegal pairings. Also, in order to handle the very large problems created by the generator, a set covering optimizer which works well for very large problems is used [4].

A typical run of the Carmen system consists of 50-100 iterations. Within each iteration and for all of the Lufthansa fleets that were tested, the generator takes 5-8 times more than the optimizer. This is partly due to the large and complex number of legality checks the generator must perform and partly due to the speed and

performance of the optimization algorithm. The memory requirements of the Carmen system are between 128 - 512MB for relatively large problems.

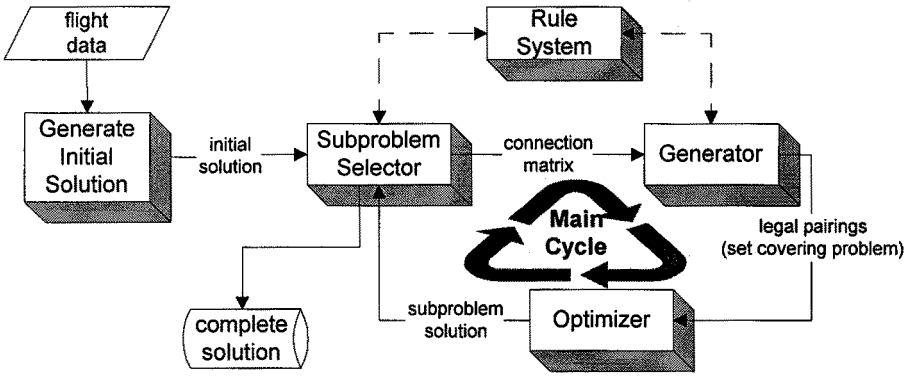

Fig. 2. Overview of the Carmen system

3 Distributed Parallel Processing

In the Carmen system the generator and the optimizer could both run faster if parallel processing techniques were used. The pairing generation phase takes, depending on the type and the size of the problem, 70-85% of the runtime required for the solution of the problem [7]. Since there exists a pairing generation algorithm which is highly parallel and fully amenable to parallel computation, we expect significant runtime reductions of the generation phase when a large number of networked workstations is used. Consequently, significant overall performance improvement is anticipated.

The Carmen optimization algorithm, takes 10-20% of the total runtime and it appears to be less amenable to efficient parallelization, especially on a distributed network architecture. This is due to the nature of the present optimization algorithm and the large amount of messages that must be exchanged. The parallelization of the optimizer would also allow for better use of the NOW memory resources.

Finally, 5-10% of the time is consumed in the analysis and the subproblem selection which is inherently a sequential process. This gives an upper bound to the speedup of the overall solution process equal to 10-20 irrespectively of the number processors used. In this paper we are mostly discussing aspects of the parallelization of the generator and present results that are significant for the overall performance of the Carmen system.

3.1 The Pairing Generation Algorithm

The generator creates a large number of pairings by connecting legs to each other in different combinations. One of the most basic and conceptually clear algorithm for generation process is a depth first search (also known as *forward enumeration*) in a search graph determined by the connection matrix representing all legal pairs between legs. The search always begins from a subset of legs known as *start legs*, mainly being those legs that depart from a crew base. In a similar manner *end legs* leading back to the same crew base can be identified for every start leg.

Pairing generation algorithm

S denotes an ordered sequence of legs
w(*S*) denotes the search-width, as a function of the working days covered by *S*

```
procedure generate()
  for each valid start leg l do
        search(l)

procedure search(l)
  S = S + {l}
  if S is legal then
     if S builds a complete pairing then
        output(S)
     endif
     for the w(S) first leg connections (l) of l do
        search(l)
  endif
  S=S - {l}
```

The generator search is limited by a maximum number of branches considered in each node of the search graph. This maximum number of connecting branches is known as the *search width* and its typical values range from 5-8 connections. In every step of the depth first search the created sequence is checked for legality. Illegal paths are not investigated further assuming that the rules have a monotonic behaviour. This implies that if a sequence is illegal it can not become legal by adding more legs to it. Depending on the rules, and the size and structure of the problem, the generation time in each iteration could take from a few minutes to many hours.

3.2 Parallel Computation Model

The generation for each start leg is independent of the generation for every other start leg. Thus, the generation work can be done by identical processes executing with different input data on separate machines. This is done, without incurring a high communication overhead for co-ordination and without performing a significant amount of additional work.

The parallelization of the generator is based on a master-slave approach. Figure 3, shows the parallel generator in the context of the Carmen system. The master is responsible for setting-up the virtual machine and spawning the slave processes. The hosts to be used, the distribution policy of the start legs and the granularity of the messages is specified by the user. The master distributes dynamically the start legs and the necessary additional problem information (e.g., connection matrix, search parameters) to the slaves. The slaves must generate all the legal pairings. The generated pairings are returned to the master. Timings, statistics and status (e.g., failures of tasks or hosts) are recorded by the master as well. A number of different message types are used for the co-operation of these processes.

The interaction protocol between the master and the slaves ensures that the slaves rarely wait for the master to send new start legs. This basically involves the

notification of the master that a slave is close to the end of his work. For efficiency reasons the master is composed of two processes, one that handles the start leg distribution and another that handles the collection of pairings from the slaves.

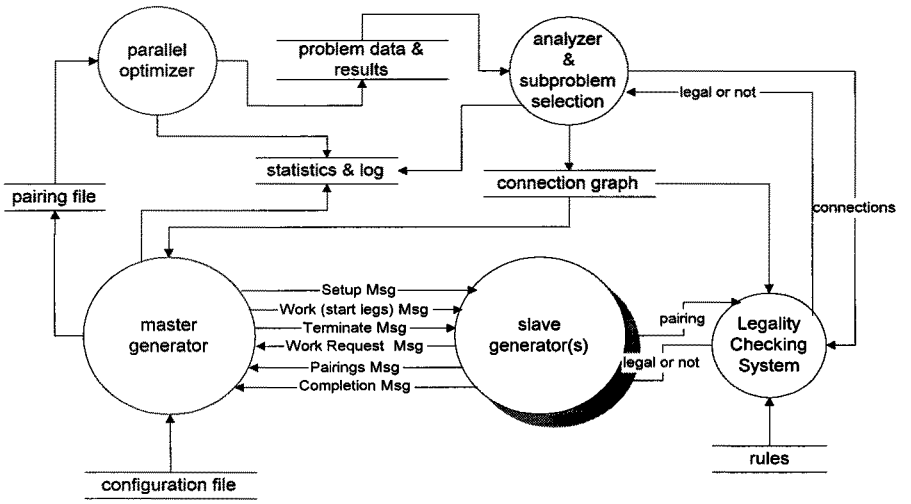

Fig. 3. DFD of the Carmen system with the parallel generator

3.3 Load Balancing

The complete distribution of all the start legs at the beginning of the run could negatively affect the load balance of the NOW. Our experience is that the search tree for each start leg is quite different, which makes it very difficult to estimate how many pairings will be generated from each start leg. In addition, another issue arises when the machines have different speeds and/or different external workloads. If not considered properly, it is possible that the whole generation process is waiting because some slave is searching its last start leg and it happens to run on the slowest machine.

A dynamic workload distribution scheme that implicitly takes into account the speed and the current load of each machine, ensures that an acceptable level of load balancing is imposed. The number of start legs sent to each slave process is dynamically determined by combining a user specified parameter and the unloaded performance index value of every machine (the sp option of the hostfile). In the beginning a sufficient number of seeds (start legs) is distributed to each slave which reduces the communication of the network. As the execution progresses the number of seeds given to the slaves decreases dynamically to one, according to a heuristic load balancing scheme that attempts to capture the actual performance of each computer.

A more complete scheme for load balancing should use dynamic load information such as CPU and memory load of each host in order to distribute the workload. This information could be provided by a commercial job management system like LSF or by the General Resource Manager of the eagerly awaited PVM version 3.4 [5].

3.4 Reliability and Fault Tolerance

The ability to reliably finish all started tasks is very important for the production version of every parallel application. In particular, this is critical for large problems which often require many hours of computation. The reliability of the generator is guaranteed using the following strategies. After the loss of communication with a slave the master cancels it from the pool of active slaves and continues its work with the remaining slaves. In addition, the loss of a slave leads to the lack of some pairings that should have been generated. These pairings must be re-computed by a new or an existing slave process. The recalculation does cause the possibility of duplication for some pairings, which is not in itself an issue since the optimizer pre-processor can eliminate them. In the worst case this involves the pairings of a single start leg.

The notification mechanism of PVM [6] is used in order to provide application level fault tolerance to the generator. The master process keeps for each task the current computing state and necessary data that in case of a slave failure can be used for the restarting step. In case of a task failure the master tries to spawn a new slave process in the same host and reassigns the unfinished part of the work to it. If this fails, it continues with the rest of the slaves and uses an existing slave process for the reassignment. Also, periodically the master checks the response time of each slave and if it exceeds a user defined response limit it treats this as a failed slave. The notification mechanism can be also used to add new slave processes to the present pool of slave processes, when a new host has been added to the virtual machine.

The system is more vulnerable to failures of the master process. In a production environment master process failures could be resolved by restarting the master on a new machine and continuing with the unfinished start legs by resetting all the slaves.

4 Performance Evaluation and Results

Profiling analysis of the serial Carmen system has shown that for medium and large schedules, the majority of the CPU time is spent in the generation phase. For one iteration of the solution process for such a problem, the sequential generator produces around 65 pairings/sec on a high end workstation [8]. With the current pairing format each pairing requires about 100 bytes to encode. Thus data is produced from the generator at approximately 6.4 KB/sec. The implementation uses NFS and local disks for I/O operations which makes the generator CPU bound. Standard Ethernet provides a maximum bandwidth of 1 MB/s, thus the parallel generator on a standard Ethernet network is also CPU bound after its start-up phase (distribution of initial data) for up to about 200 workstations.

Table 1 and 2 present problem descriptions and speedup results obtained with the PAROS parallel generator for a number of typical Lufthansa scheduling problems of various sizes. The standard production level ruleset was used for checking the legality of the pairings. Usually 50-100 subproblems must be iteratively solved for a complete solution to be found. The times shown here correspond to a typical such subproblem.

All experiments have been performed on a network of HP 715/100 workstations of roughly equivalent performance (2.89 SPECint95), connected by Ethernet at the computing center of the University of Patras. The values presented were obtained

during nights when almost exclusive usage of the network and the workstations was possible. For the serial run we have used the fastest machine in the cluster.

Problem name	No of start legs	Set-up overhead	Generated pairings (1 loop)
LH1	1366	1.2 sec	43932
LH2	2984	2.5 sec	97075
LH3	5968	4 sec	194150

Table 1. Examples of Lufthansa scheduling problems

	No of Hosts	1	2	4	5	7	8	10
LH1	Elapsed time	34.51	18.18	8.81	7.34	5.18	4.54	3.55
	Speedup	1	1.89	3.9	4.7	6.65	7.6	9.7
LH2	Elapsed time	76.64	39.80	20.23	16.13	11.4	10.19	7.98
	Speedup	1	1.92	3.8	4.75	6.72	7.52	9.6
LH3	Elapsed time	153.6	79.65	39.23	32.01	22.63	20.22	16.0
	Speedup	1	1.92	3.91	4.8	6.79	7.6	9.6

Table 2. Elapsed time (minutes) and speedup for a typical loop of the parallel generator

At start-up the legality system must be also initialized. This does take a few seconds (20-25 seconds in our experiments) but this is done only once in the first loop, and is indeed very small compared to the overall computation. The set-up overhead in each loop of the generator depends on the size of the problem (Table 1). Combining these two overheads there is a remaining serial part of less than 0.001. Thus a large number of machines can be efficiently used for the parallel generator, provided, of course, that the size of the problem is sufficient.

Even from this limited experiment it is obvious that a nearly linear reduction of the generation runtime is possible due to the highly parallelizable leg search procedure. The parallel efficiency is very high (Figure 4) and the major overhead is caused by the message passing time and the administration of the distributed system.

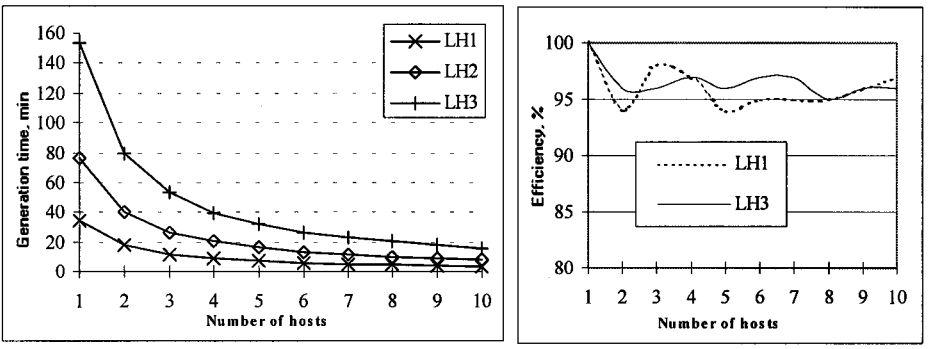

Fig. 4. Generation time (min) and efficiency of the parallel generator (1 loop)

Given that the Carmen system generator, which is currently in production at Lufthansa, requires about 80% of the total system runtime, the overall performance will be significantly improved. For example, a crew scheduling problem that requires

20 hours to solve completely, would take about 5.65 hours with 10 identical machines and 4.9 hours with 20 machines assuming that everything else remained as in the current Carmen system. Faster solution times translate to higher productivity of the airline crew management department since crew schedules can be created closer to the actual day of operation which is very important in the new deregulated environment in Europe and elsewhere. In addition, the high performance of the parallel generator allows larger problems to be solved since in the sequential system the slow runtime was the constraining factor.

5 Conclusions

Crew scheduling is one of the major problems that an airline company must solve. It is time consuming and deadline driven. In this paper a solution strategy and specific results were presented that improve and enhance the Carmen crew scheduling product with the use of parallel processing on a NOW. This architecture is available in most airlines which makes this approach highly cost effective. A significant improvement of the performance is realized by parallelizing the dominant part of the system, the pairing generator. Using the PVM message passing system we have built an efficient, robust, fault tolerant and scaleable distributed application. A large number of machines can be used efficiently without the need of excessive network bandwidth. An almost linear speedup of the parallel generation is achieved.

The improvements in performance translate immediately in significant financial and strategic benefits for the airlines. The ability to start the planning process as close to the day of operation as possible allows for fast market reactions and reduces the need for continuous rescheduling due to unexpected aircraft assignment and other schedule changes.

References

1. G.W. Graves, R.D. McBride, and I. Gershkoff. Flight Crew Scheduling. Management Science, vol. 39, no. 6, pp. 736-745.
2. E. Anderson, E. Housos, N. Kohl, and D. Wedelin. Crew Pairing Optimization. OR in the airline industry, Kluwer Academic Publishers.
3. E. Housos, T. Elmroth. Automatic Subproblem Optimization for Airline Crew Scheduling. Interfaces 27:4, July-August 1997.
4. D. Wedelin. An algorithm for large scale 0-1 integer programming with application to airline crew scheduling. Annals of Operations Research, vol. 57., pp. 283-301.
5. G.E. Fagg, K.S. London, and J. Dongarra. Taskers and General Resourec Managers: PVM Supporting DCE Process Management. Proceedings of EuroPVM 96, pp.180-187 Munich, Germany, October 1996.
6. G. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Mancheck, and V. Sunderam. PVM: Parallel Virtual Machine. MIT Press, 1994
7. Deutsche Lufthansa AG NE 4. User Requirements. ESPRIT PROJECT EP20115 - PAROS Technical Report D2.1, Frankfurt, Germany, October 1996.
8. Carmen Systems. Pairing Generation Prototype. ESPRIT PROJECT EP20115 - PAROS Technical Report D4.3 Gothenburg, Sweden, January 1997.