Integrated Computer-Aided Design System Software of Navigation Complex

Victor Sineglazov
National Aviation University
Kyiv, Ukraine
svm@nau.edu.ua

Andrew Godny
National Aviation University
Kyiv, Ukraine
andrewgodny@gmail.com

***Abstract*—Presented scheduling mechanisms for computer-aided design system for unmanned aerial vehicles with an integrated environment introduces a new approach to managing hardware resources and design time management. Usage of this mechanism allows to improve scheduling efficiency of multi-tasking applications with shared resources by allowing simultaneous operations with multiple shared data task-readers.**

Keywords—unmanned aerial vehicles; dynamic integration; computer-aided design; integrated environment; design; scheduling mechanism.

I. Introduction

Nowadays unmanned aerial vehicles (UAV) have become an essential part in every aspect of over lives. UAV are used in all sectors of production. In addition, more than 50 countries are using them. Due to the high demand for such devices, it is necessary to develop new methods for designing drones. New approach will decrease time used to develop new UAVs and significantly reduce the cost of the final product.

Let us consider the possibility of developing software tools that ensure the economic integration of relational data on the proposed method of computer-aided design environment [1]. To this end, we developed a set of software tools, consisting of a control processor, coprocessor and thematic performing processors. For the convenience of a software implementation, the control processor is presented as a server node. Thematic coprocessors are grouped as a means of dynamic data integration. Separately considered auxiliary software: drivers (D), library operations (UO DB) and data library manager.

Computer-aided design (CAD) system using the method of dynamic data integration is a structure consisting of a control processor (CP) and thematic co-processors: graphic coprocessor (GP), table coprocessor (TbP), math coprocessor (MP) and text coprocessor (TP) (Fig. 1). The design process is ensured by the design scenario of the Control processor. Designing scenario is a set of generic operations that can be represented as a graph. Generic operation consists of multiple commands for thematic co-processors with a common semantic completeness.

Since the developed system has to guarantee the optimal use of computing resources and to ensure a minimum command processing time, system that organizes the task scheduler is required, which will be responsible for compliance with the required criteria of the system.

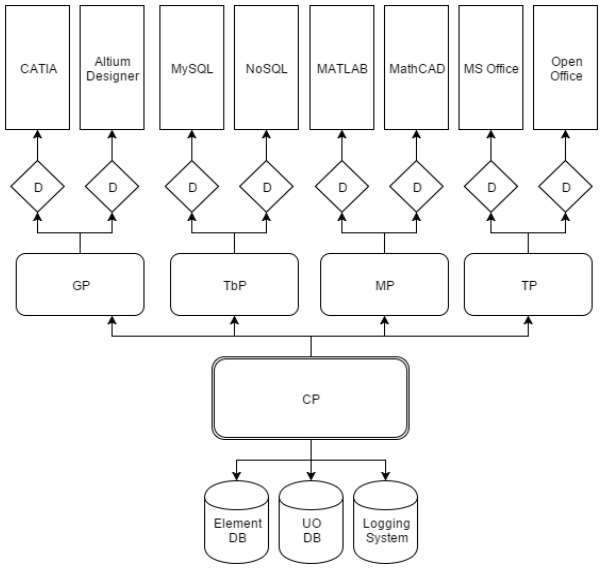

Fig.1. Computer-aided design system using the method of dynamic data integration.

Mechanisms of planning tasks - an integral part of integrable CAD, largely determines the efficiency of the use of hardware resources. Different classes of systems require different criteria to determine the efficiency of different algorithms [5]. The specifics of the planning process in the CAD defined by the requirement of timely execution of applications.

Mechanisms for CAD scheduling must specify the execution order of tasks, ensuring timely execution of design tasks with limited resources. The need to comply with deadlines brings to the fore such scheduler property as predictability [3]. This means that at any time the execution order of tasks determined by the scheduler must be unambiguous.

II. Problem Statement

Computer-aided design scheduling mechanisms have a specific performance criterion that determines the degree of enforcement of time limits problems [2]. A more efficient scheduler ensures execution of more tasks than less effective. Hence the definition: the optimal scheduling algorithm is an algorithm that provides execution of tasks with strict time constraints whenever possible. Or, in other words, if the order of the tasks defined by the optimal algorithm, leads to disruption of deadlines tasks, no other algorithm can ensure the timely execution of all tasks.

978-1-5090-1052-3/16/$31.00 ©

Computer-aided design scheduler has two components: scheduler for development period (off-line) and a scheduler for run-time (on-line, run-time) [4]. At stage of CAD development developer has information about applications and the structure of their interaction. Planning of development period is about handling of this information prior to the start of the system [2]. During the development phase processing of available information can significantly reduce the costs of planning the execution period. Scheduler is a run-time component of the system being developed and starts working on its startup; it uses the information obtained as a result of the development scheduling period.

The total amount of work performed by the two components of the scheduler remains unchanged when switching from one plan to another. It changes the distribution of work between the schedulers during the development and execution. This distribution of work is one of the main features of the classification of real-time scheduler.

Planning the development period often includes a feasibility study (schedulability analysis, feasibility analysis), the need for which arises in the case where the limits on the performance of system tasks clearly defined, and the violation of the timing of the results of calculations can lead to a corrupted system. Successful completion of the feasibility analysis ensures that when any possible load of all tasks to be completed on time.

Studies have shown that almost all systems comprising interacting tasks, accurate method of feasibility analysis is NP-hard task. Therefore, most often in practice, special mechanisms of interaction between tasks and methods of analysis are used, which gives a positive result for the feasibility of real-time applications.

When designing a scheduler for integrated CAD is necessary to solve the following problems: interlocking of shared resource, multiple lock of high priority tasks with a lower priority, composite blocking, etc. In order to solve these problems priority inheritance protocols have been developed that describe the algorithm of the scheduler based on problems mentioned above [5]. But each of these protocols has many drawbacks as they are designed for a wide range of tasks. It is necessary to develop a protocol specifically for the integrated CAD, which avoids the disadvantages of general purpose protocols.

III. Asymmetrical Priority Inheritance Protocol

Operation of synchronizing mechanism, implementing the principle of priorities inheritance, characterized by a high degree of predictability. However, the cost of predictability is excessive strictness in defining access rights for applied problems to shared resources.

The properties of the protocol. High priority task can be blocked by the task with a lower priority in the two cases. Firstly, it is apparent case of blocking, the situation in which a high-priority task trying to capture the shared resource, captured by task with low priority for incompatible operations. Secondly, it is an indirect blocking situation in which the task with medium priority is blocked by task with lower priority which inherited priority from higher priority tasks.

Asymmetrical priority inheritance protocol (APIP) does not exclude the possibility of a deadlock – a situation in which a directed graph problems blocking relationship has a cycle. Also it does not exclude multiple block, task execution may be repeatedly locked, both explicitly and indirectly. In the worst case, the number of blockings will be equal to the number of resources used, both the task itself and other tasks with higher priority.

Deadlocks can be eliminated through the introduction of a uniform procedure for the capture of shared resources, which has no cycles. However, the problem of multiple locks is not so easily solved.

Execution of high-priority tasks can be blocked during access to each shared resource, but the time of explicit blocking of task-writer with high priority on the resource may be equal to the sum of the lengths of several critical sections of lower priority tasks-readers. Generation of a high priority task-writer could be preceded by a series of captures by low priority tasks of shared resource. In this case the task-writer is forced to wait for the fulfillment of all critical sections of tasks-readers.

This problem is related to the endless waiting, but not as serious, priority inheritance effect ensures that the number of critical sections of tasks-readers with lower priority, which are blocking task-writer with higher priority, will not exceed the number of low-priority tasks-readers. This is the third problem, which exists in systems using APIP. The possibility of its occurrence will be called the problem of the composite block. Composite blocking problem causes an increase blocking time in the worst case, which may cause a decrease in the efficiency of planning.

In some cases, this drawback pays off by the ability to work simultaneously with the data shared by multiple tasks-readers. Ability to overlap in time critical sections of multiple tasks readers reduces task blocking time in the worst case, that can increase scheduling efficiency.

Most common asymmetrical priority inheritance protocol has three significant drawbacks: deadlocks, multiple and composite blocks. In some cases, APIP provides less efficient scheduling than the original PIP. The reason for reducing the effectiveness of scheduling is a possibility of composite blocks.

The use APIP can, in some cases, improve scheduling efficiency of multi-tasking applications with shared resources by allowing simultaneous operations with multiple shared data task-readers. Therefore, it is impossible to determine unequivocally which of the priority inheritance protocol is more efficient, original or asymmetrical.

IV. Asymmetrical Ceil Priority Protocol

The asymmetrical ceil priority protocol (ACPP) based on the idea of separation of references to a shared resource on the inverse for reading and writing of shared data.

As a means of controlling the inversion of priorities ACPP uses priority inheritance idea. Use of ceil priorities leads to the emergence of a new type of blocking - blocking by ceil priorities. Solely due to the addition of this type of blocking is achieved the beneficial properties of the protocol discussed below.

The first four provisions of the protocol implicitly define the order of capture of shared resources and, thus, eliminate the possibility of system failures due to deadlock [2]. In addition, the task with a high priority can enter in the first critical section not earlier than the low priority task will free all the resources it needs, therefore, task execution due to allocation of a shared resource can be postponed only once, that is no multiple blocking. Indeed, the beneficial properties are provided exclusively through the introduction of a mechanism of blocking by ceil priorities. At the same time, method of calculation of the ceil priority value does not matter. Consequently, ACPP eliminates multiple blocks and deadlocks.

Using ACPP allows simultaneous access to the shared resource by multiple tasks-readers, with the blocking compound is excluded.

Additional useful property of ACPP is achieved by adding to the parameters of shared resources second ceil priority - ceil priority for tasks-readers. Ceil priorities for readers are optional, since the value of ceil priorities of writers always matches the value of ceil resource priorities. The value of ceil priorities for readers selected so as to block all tasks writers, that is in some cases the value of the ceil priority of readers will be less than the value of ceil priority of writers. At the same time, tasks-readers, that have higher priority than task-writers will not be blocked by other tasks-readers, which may lead to overlap in time execution of critical sections of several tasks-readers. On the other hand, the ceil priorities for readers are blocking tasks-readers whose priorities are lower than priorities of tasks-writers. Due to this problem of a composite block is solved.

The asymmetrical ceil priority protocol allows simultaneous access to the same resource, for one low-priority and several high-priority tasks-readers. Thanks to this feature ACPP provides greater planning efficiency.

V. Asymmetric Preventive Inheritance Priority Protocol

Using the ideas embodied in the PIP, modifications APIP, APIPP also leads to a positive result. APIPP has much in common with the PIP. Therefore, asymmetric PIPP (APIPP) will be considered at a reduced level.

A. Determination of Protocol

Operation of synchronizing mechanism, implemented in APIPP, characterized by the following provisions:

Each resource is assigned with two threshold priorities: a threshold priority of readers and threshold priority of writers.

Ceil priority for readers used as meeting the objectives of readers' requests and is numerically equal to the priority of the task with the highest priority of those tasks that can capture this resource for writing:

$$ceil\_read = \max_{\{i|r \in modifield\_by(i)\}} pri_i. \qquad (1)$$

Ceil priority for writers is used while satisfying the query of task-writer and is numerically equal to the priority of the task with the highest priority of those tasks that can that can capture this resource for reading:

$$ceil\_write_r = \max_{\{i|r \in used\_by(i)\}} pri_i. \qquad (2)$$

Task τ, which has the highest priority among all the active tasks, takes control. Before entering the critical section in relation to the resource *r*, task τ must capture the resource for reading, if it does not modify data, or for writing, if it would modify the data.

Task τ is performed with a base priority only if it has no shared resources. Otherwise, its priority is the greatest ceil priority among all ceil priorities captured its shared resources:

$$effective\_pri_r = \max_{\{r,t|(r,t) \in got\_by(i)\}} ceil_{r,t}. \qquad (3)$$

When shared resources are released task τ gets base priority back.

Task τ_1 can supplant task τ_2 only if τ_1 priority strictly greater than the effective priority τ_2.

The task can't be completed or voluntarily suspend execution until the release of all occupied resources.

Critical sections are nested, i.e., shared resources are released in reverse order to their capture (stack).

B. The properties of the protocol

The properties APIPP coincide with those of the original PIP and APIP. Therefore, we confine ourselves to the following list of APIPP advantages:

1) APIPP eliminates the possibility of deadlocks.
2) APIPP eliminates multiple blocks.
3) APIPP eliminates composite blocks.
4) APIPP reduces the number of task switches.
5) APIPP allows tasks to be performed in one stack mode.
6) APIPP suitable for synchronizing with interrupt handlers.
7) APIPP more effective than PIP and APIP.

For these reasons, in practice it is preferred to use APIPP.

C. Example of using the protocol

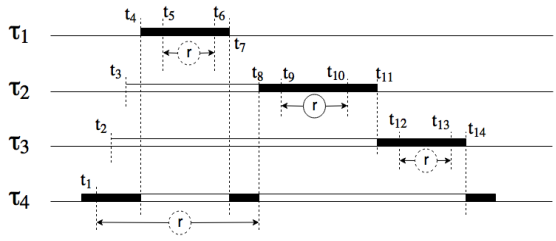

Fig. 2. Example of APIPP.

There are four tasks in the system: τ_1 with the highest priority, τ_2 medium priority, τ_3 low and τ_4 the lowest

priority; and a shared resource *r*. Task τ_2 tries to get resource *r* for writing, other tasks are trying to get it for reading. Behavior of the system in the case of APIPP is shown on Fig. 2.

According APIPP, ceil priorities of resource are set as follows: ceil priority for readers *r* established at the level of the priority of task τ_2 (writer with the highest priority), ceil priority for writers *r* established at the level of the priority of task τ_1 (reader with the highest priority).

Task τ_4 gets resource *r* for reading (t_1). Then more priority-reader task is generated τ_3 (t_2). However, the task switching does not occur, because at this time priority of the task τ_4 equal to a ceil priority of the readers who captured resource, i.e. to the priority of tasks τ_2. Such a preventive blocking of tasks-readers, whose priority is less than the ceil priority of readers avoids composite blocks. The fact that the tasks which are using resource that is captured by other task, do not get control before its release (rather than blocking) reduces the number of task switches and allows to perform all tasks via single stack. Then higher priority task-writer is generated τ_2 (t_3). However, the task switch does not occur again due to the same circumstances. This way of organizing mutual exclusion mode prevents multiple blocking of tasks and even more so – eliminates the possibility of a deadlock. Next, task τ_4 is superseded by task τ_1 (t_4), which has been successfully performed (t_7) using the resource *r* (from t_5 to t_6) for reading, as the priority τ_1 is strictly greater than the ceil priority of *r* readers. As a result of this overlapping of critical sections of two tasks-readers we accomplished an increase of the efficiency of planning. Next, task τ_4 releases *r* (t_8). At this point, the most priority task among proactively blocked (τ_2) is unlocked and completed successfully (t_{11}) using the resource *r* for writing (from t_9 to t_{10}). Thereafter, control is passed to the task τ_3, which also completed successfully (τ_{14}) using the resource *r* (from τ_{12} to τ_{13}). Control again is passed to the low priority task τ_4.

VI. Conclusions

Reviewed asymmetrical priority inheritance protocol has three significant drawbacks: deadlocks, multiple and composite block. In some cases, the asymmetric priority inheritance protocol provides less efficient planning than the original priority inheritance protocol. The reason for reducing the effectiveness of the planning is a composite block.

The use of asymmetrical priority inheritance protocol can, in some cases, improve the efficiency of the planning of multi-tasking applications with shared resources by allowing simultaneous operation with multiple shared data by tasks-readers. Therefore, it is impossible to determine unequivocally which of the priority inheritance protocol is more efficient, original or asymmetrical.

The proposed asymmetric preventive inheritance priority protocol eliminates the possibility of deadlock, multiple and composite blockings. By permission of the simultaneous reading of shared data for multiple tasks, asymmetric preventive inheritance priority protocol provides more efficient planning in comparison with the original protocol, which is the most effective of the known protocols.

The proposed protocol of asymmetric preventive priorities inheritance, implementing the idea of inherent in the asymmetric protocol of ceil priorities. Its use also eliminates the possibility of deadlock, multiple and composite blocks. Due to permission of the simultaneous reading of shared resource in multitasking mode APPIP provides more efficient scheduling in comparison with the original protocols.

References

[1] K. Lee, CAD Basics (CAD/CMA/CAE), Peter Press, 2004.

[2] V.M. Synehlazov, O.I.Chumachenko, A.P. and Godny, "Information technologies of computer aided desin systems based on dynamic data integration and simulation procedures" 2 International conference "Computer Algebra and Information Technology." Odessa, August 2016, pp. 9-10.

[3] G. Berezhnoj, "Problems building large IT systems", PCworld, 1998. (in Russian)

[4] I. P. Norenkov, Basics of computer-aided design Peter Press, 2002. (in Russian)

[5] Kristi Morton. Dynamic Workload Driven Data Integration U. of Washington, 2012.