Efficient Real-time Parallel Signal Processing for Decentralized Control Using Group-Pipelined Scheduling

J. Roberts, H. Boussalis, C. Liu, J. Dong, K. Rad, P. Thienphrapa, Z. Purnajo, and S. Fallorina
California State University, Los Angeles

Abstract

This paper presents a technology of generic group-pipelined scheduling for decentralized control of large flexible structures of a segmented reflector telescope, the James Webb Space Telescope (JWST) testbed. Such a structure is controlled by a number of lower-order decentralized local controllers, leading to reduced computational complexities, and ease of development of a coarse-grain parallel control algorithm. The key algorithmic technology presented in the paper is the task scheduling of the parallel control algorithm; the decentralized control tasks have to be mapped and scheduled onto the processors to fully utilize the computational power. The group-pipelined approach, compared against a straightforward task scheduling approach, can result in perfect load balancing and fault-tolerance should one of multiple processors fail. This approach can also significantly reduce the "ripple control delay" of stabilization introduced by a previously used basic pipelined approach.

1. Introduction

As the successor to the Hubble Space Telescope (HST), the James Webb Space Telescope (JWST), formerly known as Next Generation Space Telescope (NGST), requires a larger light-gathering mirror capable of detecting faint signals [1]. Due to the manufacturing and deployment difficulties of using a monolithic piece of glass, the primary mirror of the JWST will consist of several smaller reflecting panels. However, a reflector built from segments relies on an active control system for precision alignment of the optical surface. This control system is responsible for achieving high-precision figure control and maintenance of the reflector surface to a calibrated parabolic reference figure in a dynamic disturbance environment.

To study the control of such large segmented optical systems, the National Aeronautics and Space Administration (NASA) in 1994 provided funding to establish the Structures Pointing and Control Engineering (SPACE) Laboratory at the California State University, Los Angeles (CSULA). One of the major goals of this project is to design and fabricate a testbed that resembles the complex dynamic behavior of a segmented space telescope.

The control of the SPACE testbed proceeds as follows: Unknown external forces displace the mirror's segments into incorrect positions; in the JWST this event would corrupt images received by the telescope. The inductive sensors detect these displacements and convert them into corresponding electrical signals. The analog-to-digital converters then sample these voltages and digitize such signals. Then, a decomposition technique is employed that results in physical or mathematical decentralization of the structure into lower-order subsystems [3]. The control system processes the subsystems in a decentralized fashion to produce appropriate control outputs. Such control outputs, after being converted back into continuous voltages are sent to the actuators in order to reposition the displaced panels into a correct configuration. The control system performs this control cycle iteratively to actively maintain precise mirror shaping.

The decentralized control algorithm, by nature, can be performed in parallel using multiple processors [12]. An embedded parallel architecture has been employed in the SPACE Laboratory testbed. The real-time embedded system, utilizes a Pentek 4285 board that is configured with four TMS320C40 digital signal processors [9]. Each processor has its own local memory in addition to a globally shared memory space. High-speed bidirectional communication ports allow direct message passing between processors, while the shared global random access memory (RAM) is accessed through a common bus. Here, the processors are arranged in a tree topology with one processor configured as the master since only it has access to the digital-to-analog (D/A) and analog-to-digital (A/D) converters. These signal converters, in turn, are connected to the actuator amplifiers and sensors respectively. The control of the primary mirror requires 18 sensor inputs and 18 actuator outputs; this multiple input-multiple output (MIMO) system attests to the computational requirements of this application.

In performing parallel processing we refer to the control of a single subsystem as a *control task*, or simply *task*. In each control cycle we wish to distribute the number of tasks, M, among the number of available processors, P.

Based on the decentralized model we make the following assumptions:

1. Each task is not further decomposed.
2. Computational complexities of all tasks are identical.
3. Each task completes a control cycle.
4. There are no data dependences among tasks [8].

Based on the features of the control algorithm and the tailored embedded architecture, three task scheduling schemes – straightforward, basic-pipelined, and group-pipelined – are compared in the paper. Both our analytical study and the preliminary experiments show that the basic-pipelined approach can improve the utilization of the processors, and thus, result in a higher throughput than the straightforward one. On the other hand, the basic-pipelined approach causes a delay of stabilization due to a fact of "ripple control" of the decentralized system. Ripple control is the effect of the properties of a pipeline on the control output. This effect is that all of the control variables do not update simultaneously using the basic-pipelined design. Such a delay can be alleviated by using the group-pipelined approach.

The rest of the paper is organized as follows. Section 2 describes the straightforward task scheduling. Section 3 describes the basic-pipelined approach. Section 4 describes the group-pipelined approach, and justifies the use of such an approach to alleviate the ripple control delay. Section 5 shows the experimental results and comparisons of the three approaches. Section 6 concludes the paper.

2. Straightforward Task Scheduling

Based on the system decentralization [6], one straightforward approach is to assign the tasks to the processors as evenly as possible within the same control cycle. If *M* is a perfect multiple of *P*, then perfect load balancing is achieved by evenly distributing the tasks among the processors. Otherwise, a subset of the processors will be idle during certain control cycles due to the load imbalance. Such a scenario is illustrated in Figure 1 with M=6 and P=4. In this approach the length of the control cycle must extend to the amount of time required by the slowest processors, that is, the processors with the heaviest loads. Optimal capacity is not achieved in this situation because processors with lighter loads are idle while waiting for processors with heavier loads to complete their tasks.

Another issue is that this mechanism does not lend itself favorably towards fault tolerance because the failure of one processor will result in the failure of its corresponding subsystems, an unacceptable scenario.

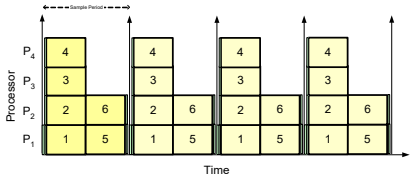

Figure 1. Straightforward scheduling (M=6 P=4)

3. Basic Pipelined Design

Capitalizing on the nature of decentralized control, a more sophisticated parallel design approach has been deployed to provide load balancing and fault tolerance. Using this approach, each processor has the capability to handle control calculations for an arbitrary subsystem. Each individual task is scheduled in a pipelined fashion among the available processors, so the sequential order of the control cycles can be observed. This approach allows perfect load balancing for any numbers of *P* and *M*. Furthermore, this technique promises to tolerate failure of one or more processors, since any one of the functioning nodes is able to control any subsystem. The task rescheduling methodology for fault-tolerance can be found in [6]. In this design, each processor keeps a copy of all of the constant matrices associated with the control loop in its local memory for all *M* controllers. These constant matrices account for the geometry of the structure. While large values of *M* require more local memory for each processor, this approach reduces the traffic on the common bus due to the accesses of the shared memory space. On the other hand, the sensor input data, state variables and the calculated control output signals are communicated via message passing.

Note that in this real-time embedded system, the control signals of a specific decentralized controller are used to trigger the actuators to move the corresponding panel. Sensor readings of panel displacements are read and represented by an input vector. The input vector is then used for the operation of the next iteration of the control output. Thus, there is an automatic serialization between the accesses of the global vectors; no racing problem can occur. Figure 2 illustrates the general parallel pipeline.

One the of key problems of the basic-pipelined approach observed is that if *M* is much larger than *P,* the decentralized control will ripple the different components of the panels, leading to a delay of stabilization [13]. Such problem has been identified from our previous research. In this paper, a group-pipelined-based approach is devised to alleviate such a phenomenon as described in the next section.

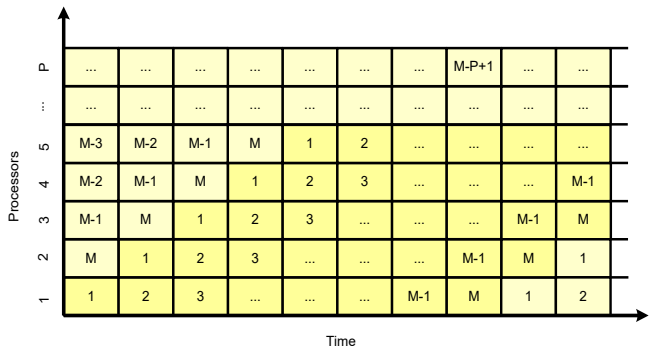

Figure 2. Generalized pipelined processing

4. Group-Pipelined Task Scheduling

In the group-pipelined approach, the controller tasks are grouped and scheduled in a pipelined fashion among the available processors, and the sequential order of its control cycles remains intact. This method allows for improved load balancing for any combination of the numbers of processors and tasks. Task scheduling and rescheduling are used to handle the cases when one or more processors fail. A similar technique can also be employed to optimize the parallel processing of tasks when failed processors are recovered.

In this paper, a group-pipelined scheduling is used to alleviate the effect of ripple control delay. The M control tasks are grouped into super-tasks with $floor(M/P)$ control tasks each. Then the super-tasks are scheduled in a pipelined fashion among the available processors as before. Using this approach, there are $floor(M/P)*P$ tasks being performed in every control cycle. Experimental results are shown to compare the performance of the group-pipelined approach against that of the straightforward and basic-pipelined approaches.

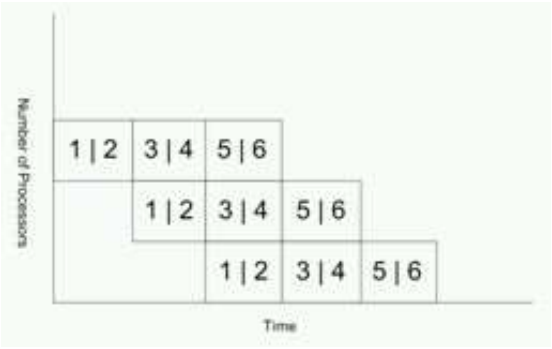

Figure 3. Grouped-pipelined task scheduling

Using the group-pipelined approach, a better linearity of throughput is observed as the number of processors increases. Also, the delay of stabilization due to the rippled control of the subsystems has been reduced since all tasks are handled simultaneously in each control cycle.

Note that, this approach still retains the features of the basic-pipelined scheme, that every processor performs all control tasks. Thus, it is easy to perform task rescheduling should a processor fail. On the other hand, since M is not necessarily a perfect multiple of P, the load balancing can be sacrificed. Figure 3 demonstrates group-pipelined task scheduling with three processors.

5. Experimental results

The straightforward parallel implementation of decentralized controllers has been realized successfully in the SPACE testbed. These decentralized control algorithm codes were written in C and implemented using up to four DSPs running in parallel.

The speedup curves shown below in Figure 4 demonstrate the effectiveness of the parallel processing. The processing time is reduced as the number of processors is increased, thus allowing the attainment of real-time control objectives. However, due to the coarse-grain nature of the decentralized controller tasks, there is no difference in speed for the cases P=3 and P=4 using the straightforward approach.

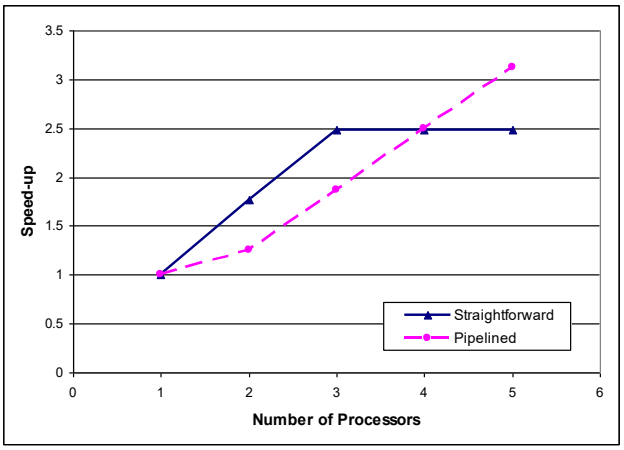

Figure 4. Speedup curves

In the cases P=1 to 3, the straightforward approach yields better speedup results than pipelining due to the overhead incurred under pipelining, which is caused by message passing and the increased repetition of data input and output. However, the pipelined task scheduling technique shows superiority in performance for the cases P=4 and 5 due to increased throughput. The results below show that pipelined scheduling exhibits competitive performance in comparison with the straightforward approach while featuring fault tolerance. The following figures provide means with which to compare the performance of the pipelined implementation and the straightforward

implementation. Figure 5 describes the straightforward implementation, while the figure 6 describes the pipelined implementation. Figure 7 shows the grouped pipeline results. It is the finding that the grouped method's performance is similar to that of the straightforward method for one, two and three processors.

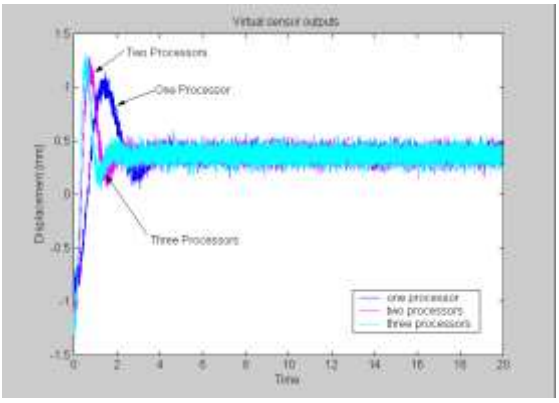

Figure 5. Straightforward implementation

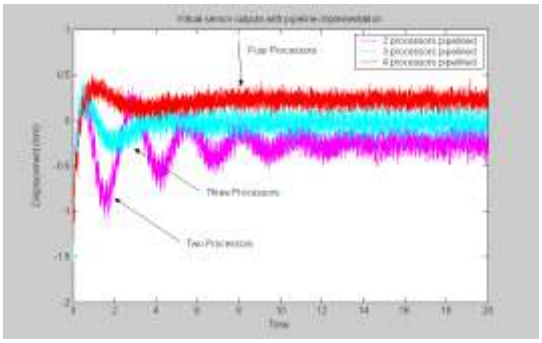

Figure 6. Pipelined implementation

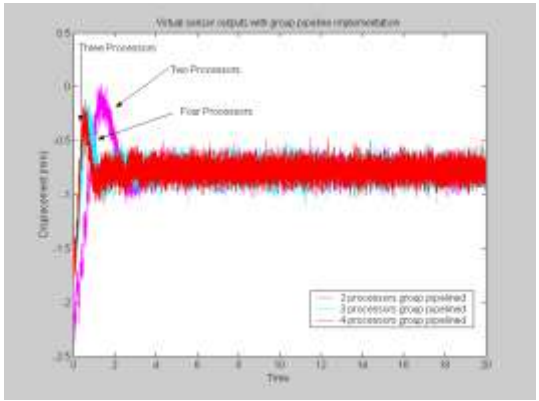

Figure 7. Grouped-pipelined task scheduling

The shape of the stabilization plot in the straightforward case is the same for different numbers of processors because all six subsystems are processed in every control cycle. On the other hand, stabilization under pipelining exhibits different ripple delays for different numbers of processors. This is likely the result of different numbers of subsystems being processed per control cycle, depending on the number of processors present. Particularly in the two-processor basic-pipelined scenario, the system appears to stabilize slowly due to an underdamped response, as only two subsystems are controlled per control cycle. Such underdamped behavior is not shown in the group-pipelined approach since all six subsystems can be controlled in each cycle as discussed in Section 4.

6. Conclusion and Future Work

Parallel program design and realization has been implemented successfully using decentralized control algorithms. The implemented decentralized controllers have the following features: achieve desired system performance, allow the use of small memory space, reduce computational complexity, and simplify the development of parallel programs. Real-time control performance has been achieved using a straightforward parallel program design. However, such a standard approach suffers from poor load balancing and is not resilient to processor failures. By capitalizing on the natural parallel structure of decentralized control, a fault-tolerant group-pipelined parallel processing design has been developed. This approach features improved load balancing for any number of processors and tasks. Pipelined task scheduling seeks to improve the performance in the cases when M is not an integer multiple of P. Additionally the system allows recovery from one or more processor failures. Also, such an approach alleviates the delay of stabilization due to the ripple control problem introduced by our previously employed basic pipelined approach.

Future work will involve more special cases and more complicated scenarios such as handling master processor failure. Fault detection for single and multiple processor failure, reconfiguration of tasks, analysis of time data and optimization of message passing will also be accomplished.

7. Acknowledgements

This work was supported by NASA under Grant URC NCC 4158.NNG04GD69G. Special thanks go to all the faculty and students associated with the SPACE Lab.

8. References

[1] H. Stockman, *The Next Generation Space Telescope; Visiting a Time When Galaxies Were Young*. June 1997.

[2] H. Boussalis, "Decentralization of Large Space-borne Telescopes", Proceedings of SPIE Symposium on Astronomical Telescopes, 1994.

[3] H. Boussalis, M. Mirmirani, A. Chassiakos, and K. Rad, "The Use of Decentralized Control in the Design of a Large Segmented Space Reflector," Control and Structures Research Laboratory, California State University, Los Angeles, Final Report, 1996.

[4] H. Boussalis, M. Mirmirani, K. Rad., M. Morales., E. Velazquez, A.G Chassiakos, J.A Luzardo, "The Use of Decentralized Control in the Design of a Large Segmented Space Reflector," NASA URC Technical Conference, Albuquerque, NM. February, 1997.

[5] D.D. Siljak, *Decentralized Control of Complex Systems*, New York: Academic, 1991.

[6] S. Fallorina, H. Boussalis, C. Liu, K. Rad, J. Dong, A. Khoshafian, P. Thienphrapa, and Dani Nasser, "A Generic Pipelined Task Scheduling Algorithm for Fault-Tolerant Decentralized Control of a Segmented Telescope Testbed," Proceedings of ASME DETC/CIE 2004, September – October, 2004, Salt Lake City, Utah.

[7] I. Foster, *Designing and Building Parallel Programs*, Addison-Wesley Publishing Company, Inc. 1995.

[8] J. Hennessy, D. Patterson, *Computer Architecture, A Quantitative Approach*, Morgan Publishing, San Francisco, Calif., 1990.

[9] *TMS320C4x User's Guide*, Texas Instruments, Inc., 1991.

[10] *Octal TMS320C40 Processor Manual*, Pentek, 1998.

[11] J. Liu, *Real-time Systems*, Prentice-Hall, Inc. 2000.

[12] H. Boussalis, E.B. Kosmatopoulos, M. Mirmirani, P.A. Ioannou, "Adaptive Control of Multivariable Nonlinear System with Application to a Large Segmented Reflector", ACC 1998.

[13] P. Thienphrapa, S. Fallorina, H. Boussalis, C. Liu, K. Rad, J. Dong, D. Nasser, "A Generalized Fault-Tolerant Pipelined Task Scheduling for Decentralized Control of Large Segmented Systems", Proceedings of CCCT 2004, August 2004, Austin, Texas.