NASA Technical Memorandum 101519

A Strategy for Reducing Turnaround Time in Design Optimization Using a Distributed Computer System

Katherine C. Young, Sharon L. Padula, and James L. Rogers

(NASA-TM-101519) A STRATEGY FOR REDUCING TURNAROUND TIME IN DESIGN OPTIMIZATION USING A DISTRIBUTED COMPUTER SYSTEM (NASA) 10 p
CSCL 09B
N89-13991
Unclas
G3/61 0179848

October 1988

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665-5225

A STRATEGY FOR REDUCING TURNAROUND TIME IN DESIGN OPTIMIZATION USING A DISTRIBUTED COMPUTER SYSTEM

Katherine C. Young, Sharon L. Padula, and James L. Rogers

NASA Langley Research Center
Hampton, Virginia 23665

ABSTRACT

There is a need to explore methods for reducing lengthly computer turnaround or clock time associated with engineering design problems. Different strategies can be employed to reduce this turnaround time. One strategy is the use of a supercomputer, which can be costly in terms of hardware acquisition and software modification. Another strategy is to run validated analysis software on a network of existing smaller computers so that portions of the computation can be done in parallel. This paper focuses on the implementation of this second strategy using two types of problems. The first type is a traditional structural design optimization problem, which is characterized by a simple data flow and a complicated analysis. The second type of problem uses an existing computer program designed to study multilevel optimization techniques. This problem is characterized by complicated data flow and a simple analysis. The paper shows that distributed computing can be a viable means for reducing computational turnaround time for engineering design problems that lend themselves to decomposition. Parallel computing can be accomplished with a minimal cost in terms of hardware and software.

INTRODUCTION

Traditionally, large aerospace design problems are divided into disciplines with each discipline contributing to the design of one or more components and to the configuration of the entire vehicle. This division of labor simplifies the task of individual design teams and allows them to make progress even when physically separated from one another. The division of large design problems into subsystems also avoids some of the limitations that computers have in handling the total structural analyses and design optimization of large problems.

In recent years there has been promising research done in the field of multilevel optimization for large structural design problems.[1,2] Using multilevel optimization methods, large problems are decomposed into hierarchically related smaller subsystems, where the structural analysis and design optimization for each subsystem are done simultaneously.

The multilevel optimization method relates well to current research in computer science in the areas of parallel processing[3] on a parallel computer and distributed processing over a network of computers. If optimization for the subsystems can be run simultaneously, then turnaround time for the total system optimization can be reduced. While parallel processing is a means of implementing multilevel decomposition techniques, this option generally requires hardware acquisition and computer code modification. A less costly option is to use an existing network of computers to simulate parallel processing. It is, therefore, the purpose of this paper to investigate distributed computing over a network of existing workstations as a means of decreasing the turnaround time for design optimization problems. Distributed computing will be applied to two types of problems. The first problem chosen is a traditional structural design optimization problem, which is characterized by simple data flow and complicated analysis. The second problem chosen is an idealized design optimization problem that utilizes multilevel optimization techniques. This problem is characterized by complicated data flow and simple analysis.

AVAILABLE COMPUTER RESOURCES

The hardware chosen for this project is a network of eight MicroVAX* workstations and an Ethernet local area network(LAN) circuit (Figure 1). Each MicroVax workstation is configured with 6 million bytes (6MB) of main memory and a 71MB hard disk.

Although eight workstations are available for distributed computing, only a subset of these is used. This is done for several reasons. First, experience has shown that hardware problems occur and requiring all eight of the workstations to be operational at all times is unrealistic. Second, the primary user of a workstation often has a CPU intensive program running, which inhibits the execution of the distributed batch processes. Therefore, the workload for a workstation is examined before it is chosen to be used in the distributed system.

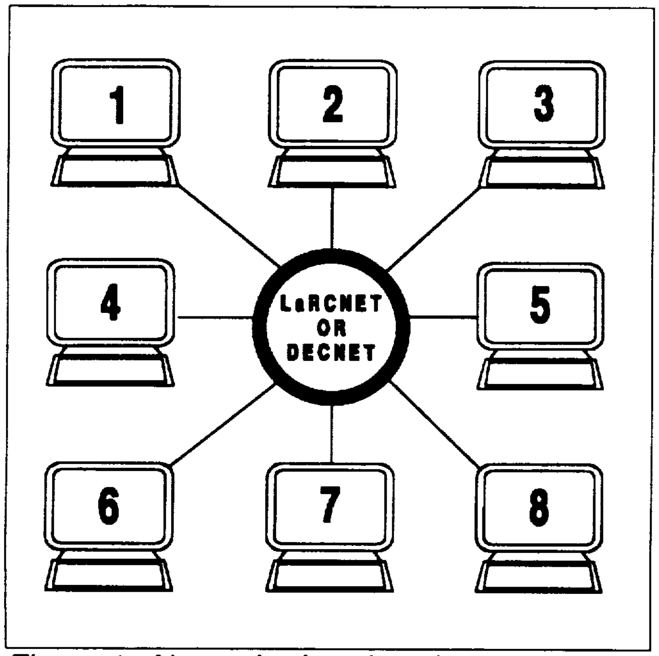

Figure 1. Network of workstations

The workstations are connected by two networks, DECnet and LaRCNET. DECnet is the product name for software and hardware that allows Digital Equipment Corporation (DEC) computers to operate over a communications network. LaRCNET[4] is a local area network developed at the NASA Langley Research Center (LaRC) to provide file transfer between multi-vendor distributed computers at the Langley site. Both networks use the same physical Ethernet, but because of software implementation differences in accessing the network, LaRCNET's transmission rate is faster than DECnet's. However, because LaRCNET sometimes experiences problems and is not operational, a choice of network is made at execution time. The software selected for this project consists of application programs written in FORTRAN 77, and the DEC VAX/VMS[5] operating system, which allows the use of remote batch processing. The two application programs are described next in this paper.

*Use of commercial products and names of manufacturers in this report does not constitute an official endorsement of such products or manufacturers, either express or implied, by the National Aeronautics and Space Administration.

DISTRIBUTED STRUCTURAL OPTIMIZATION

In 1986 engineers at Langley Research Center were given the task of redesigning the solid rocket booster (SRB) joint that had failed in the Shuttle tragedy. One group of engineers at LaRC was involved in optimizing a new design that might be a candidate for the booster joint if the existing SRB joint could not be successfully modified[6,7].

The SRB joint redesign is a structural sizing problem whose goal is to reduce the weight and stresses in the joint. PROSSS[8], a system of structural analysis and optimization computer programs is used in this design optimization work which is done on a DEC MicroVAX workstation. Seven design variables are used in the optimization process. To estimate the gradients of the objective function and constraints with respect to a given design variable, that variable is perturbed and the finite element model is reanalyzed. The analysis of the finite element model is repeated eight times, once for each design variable and once for the baseline. A single optimization cycle (seven design variables and the baseline analysis) takes three hours and forty five minutes to complete. The time spent by the optimization software is negligible compared to the time spent in analysis. Since five to seven cycles are needed to converge to final optimization results, the turnaround time is from nineteen to twenty six hours. Because of the need to reduce turnaround time, it is expedient to distribute the SRB joint design optimization system over a network of workstations.

Since all eight of the workstations used in this project are seldom available at one time, the system is distributed to some set of four. Figure 2 shows the mapping of the problem onto four workstations. Three of these workstations run only the analyses for the perturbed design varimodifications are required in PROSSS to allow the analysis for the design variables to be distributed.

The parallel processing for this distributed system is initiated by a set of five VAX/VMS DEC command lan-

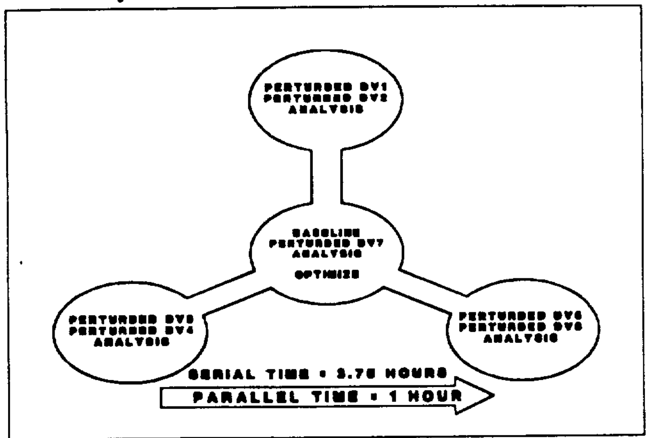

Figure2. Four workstation distribution for SRB joint opt

guage (DCL) files: one interactive procedure and four batch procedures. The interactive procedure queries the user for input parameters such as: total number of design variables, maximum number of optimization cycles, names of the workstations to use, which network software to use, and the time limit to wait for the data file to be received. The interactive procedure also submits the batch procedure files to queues on the distributed workstations. The four batch procedures control the data flow between the workstations. Their purpose is to monitor data files, execute the analysis, and then send the results to the next step of analysis. These batch procedures also contain instructions to close down the distributed system if a data file does not arrive within a specified time limit. These checks are necessary because of the possibility of hardware and software problems. It is observed that if one of the workstations procedures terminates abnormally, then the other workstations wait indefinitely and large event logging files are generated which eventually fill these workstation disks.

This design problem has the necessary computational requirements (long analysis times and uncomplicated data flow) to demonstrate the value of a distributed computer system. The resulting distributed system reduces the time of one optimization cycle from three hours and forty five minutes to one hour. This work is documented in reference 6.

DISTRIBUTED MULTILEVEL OPTIMIZATION

As aerospace systems become more complex, the need for automatic and mathematically rigorous system integration becomes more critical. Multilevel optimization is one technique for integrating component designs into a total system design and then iteratively improving the performance of the system as a whole. Multilevel optimization can be hampered by the sheer size of the problems involved. For example, optimizing the design of a SRB joint component taxes micro-computer resources; combining all design components into an optimum solid rocket booster model can overwhelm the largest computer available.

Application of distributed processing is another way to improve the performance of system integration techniques. Typically, the most complicated system, with hundreds of components and thousands of design variables, has relatively small amounts of data coupling. This means that the components can be designed separately and in parallel. Only gross details and the sensitivity of those solutions to changes in other parts of the system must be communicated. Thus, the same type of parallel processing which is so effective in the SRB joint design should be beneficial in large design problems.

The Multilevel Simulator

The multilevel simulator is a computer program[9] designed to investigate multilevel optimization techniques. The simulator mimics the qualitative behavior and data couplings occurring among subsystems of a complex engineering system. Simple analytical functions are used in place of realistic disciplinary analyses. In this way, numerous multilevel optimization techniques can be investigated in a relatively short period of time.

Converting the multilevel simulator for distributed processing is considerably more complicated than for the SRB joint design problem. System design problems often decompose into numerous levels. For example, an aircraft is composed of different subsystems such as propulsion, wings, fuselage, etc. The propulsion subsystem is also a collection of subsystems such as compressor and turbine. Thus, the aircraft system has at least three levels. The multilevel simulator allows decomposition to any number of levels with any number of subsystems on a level.

The original multilevel simulator is a single FORTRAN computer program. Information is passed from one subsystem to another as parameters in subroutine calls. The distributed multilevel simulator requires the conversion of subroutines into distributed computer programs. Minor software changes are required in the input and output sections of the distributed programs. Each subsystem needs to receive data from a lower level, execute its optimization code and then pass the computed results to the next higher level. In addition to changes in the simulator programs to allow for input and output of data, one new FORTRAN program is needed to act as a data manager. The data manager collects the data from the subsystems on one level, and then transfers the combined data to each of the subsystems on the next level. Each level has a continuously executing data manager program.

The test case chosen has 25 design variables and is decomposed into four levels with a total of nine units (one system and eight subsystems) (figure 3). To allow the work of the simulator to be distributed to different processors requires the partitioning of the multilevel simulator application program. Nine copies of the optimization code and related analysis codes are required for the test case. The distributed applications codes are denoted by TOPLEVEL and LEViSj, where i denotes level number and j denotes subsystem number. Thus LEV1S1 is the code for Level 1 subsystem 1, LEV2S1 is the code for Level 2 subsystem 1 and so on.

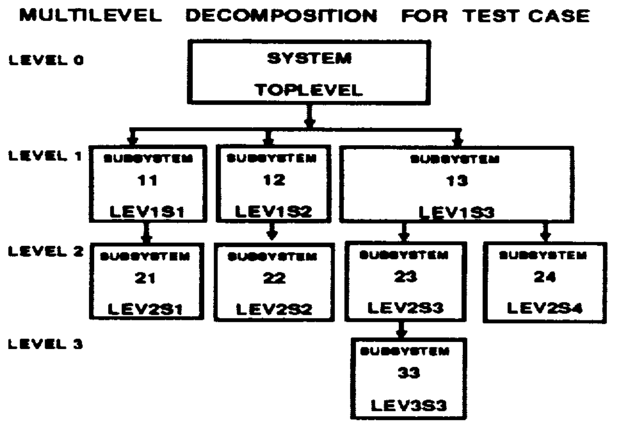

Figure 3. Schematic of multilevel simulator

Five workstations are used for distributing the subsystems (figure 4). Of the five workstations used, one is the controller workstation on which the system level and data manager codes are executed and four handle the sublevels and their analyses. Each workstation handles all levels for the same subsystem. For example as shown in Figures 3 and 4, LEV1S3, LEV2S3, and LEV3S3 are grouped together on the same workstation. This is facilitated in VAX/VMS by making a directory or separate work area for each level on the workstation. With the five workstation hardware configuration, any number of levels can be handled with the number of subsystems per level limited to four.

DCL Command Files

After partitioning the FORTRAN application program into independent program units, the next task is to adapt the command files that distribute and control the execution of the system. These command files are written in the VAX/VMS DEC Command Language (DCL)[5]. There are DCL command files for each subsystem (8), each data manager (2) and the total system (2), resulting is a total of twelve for the test case. Of these twelve DCL files, there are five types (see Appendix for a listing). Type I is an interactive procedure and Types II, III, IV and V are batch processors that are started by the interactive procedure and continue to execute until a termination file appears in their directory or work area.

The interactive procedure, Type I, is similar to the interactive procedure for the distributed SRB joint problem. It queries the user for input parameters as to which network to use, the time limit that a workstation should wait

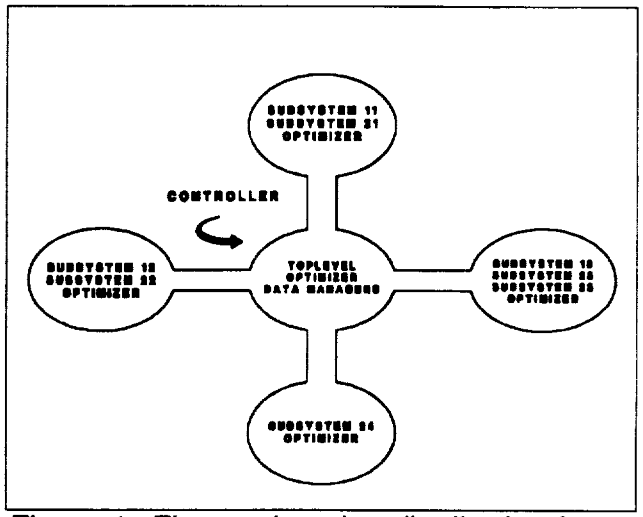

Figure 4. Five workstation distribution for simulator

for data to arrive, number of levels and subsystems, which workstations are to be used in the distribution, and which workstation is acting as the controller. This command file also distributes the FORTRAN executable file, input data file and a DCL command file to each workstation for each subsystem executing on that workstation. Its final function is to submit the DCL command file to the batch queue of each distributed workstation for execution.

Type II is a batch procedure file that waits for the required output data files from the other subsystems on that same level, executes the data manager for that level, and then passes the combined data up to the subsystems on the next level as input files. There are two such procedure files, one for level 2 and one for level 3.

A Type III procedure file executes each subsystem and is the simplest of the three procedure files. Its function is to wait for a data input file, execute the optimization code and pass the output to the level data manager.

the results down to the lowest level. If convergence has occurred or the maximum number of cycles has been met, the top level optimizer generates an output file. This output file acts as a signal to the time checker procedure (Type V) to send the termination file to all workstations.

Type V is a procedure that is used as the time checker and system terminator. The interactive procedure (Type I) passes the time limit, and workstation names to this procedure as parameters. When the time limit is exceeded or the top level optimizer generates an output file, a termination file is sent to all workstations.

Table 1

Problem	#Levels	#Design Variables	Turnaround in min	File Transfer
Simulator Distributed	4	25	7	33
Simulator Sequential	4	25	2	0
SRB Distributed	1	7	60	18
SRB Sequential	1	7	225	0

RESULTS

Table 1 summarizes the difference between the distributed and sequential version of both the SRB joint redesign and multilevel simulator problems. Notice that the turnaround time for the distributed simulator is longer than that for the original sequential program. That is to be expected because the simulator test case has many levels, many design variables and lots of files to transfer, but has very simple analysis requirements. The data transfer requirements for the distributed simulator test case simply overwhelm the problem and explain why the distributed case takes over three times more clock time than the sequential case. If this was an actual system design problem instead of a simulated one, the cost of data transfer would probably be negligible compared to the cost of analysis. For example, utilizing distributed computing in a component design problem like the SRB joint design produces considerable savings in turnaround time. Here the number of levels and design variables is small, the number of files transferred is moderate, but the analysis time is large.

CONCLUDING REMARKS

It is concluded that using a distributed computer system of existing hardware, linked by a network, is an effective way to reduce turnaround time. The benefits of parallel computing can be simulated with minimal changes to application software. Experimentation with the distributed and serial implementations of the multilevel simulator reveals that the time required for data transmission from one network workstation to another may exceed the total turnaround time savings if the analyses performed at each workstation have short execution times. On the other hand, the distributed implementation of the SRB joint problem proves to be beneficial because the analysis time is extensive. Thus, the conclusion is reached that the distributed system is workable and will reduce turnaround time for typical engineering design problems.

REFERENCES

1. Sobieszczanski-Sobieski,J.; James, B. B.; and Riley, M. F.: "Structural Sizing by Generalized, Multilevel Optimization." AIAA Journal, Vol. 25, Number 1, January 1987, pp. 139-145.
2. Wrenn, G. A., and Dovi, A. R.: "Multilevel Decomposition Approach to the Preliminary Sizing of a Transport Aircraft Wing"; AIAA/ASME/ASCE/AHS 28th Structures, Dynamics, and Materials Conference, Monterey, Cal., April 6-8, 1987, AIAA Paper No. 87-0714-CP.
3. Noor, A. K.; Storaasli, O. O., and Fulton, R. E.: "Impact of New Computing Systems on Finite Element Computations", State-of-the-Art Surveys on Finite Element Technology, edited by Noor and Pilkey, ASME Special Publication H00290, Nov. 1983, pp. 499-530.
4. Riddle, E. P.: "NASA LaRC Distributed Computing Network", Vim 44 Conference Proceedings, April 1986, pp. 2-44 through 2-49.
5. MicroVMS User's Manual, Digital Equipment Corporation, Order Number AA-FW63A-TN, Part 2, June 1985.
6. Barthelemy, J. F. M.; Chang, K.; and Rogers, J. L. Jr.: "Shuttle Solid Rocket Booster Bolted Field Joint Optimization". AIAA Paper 87-0702-CP, Presented at the AIAA/ASME/ASCE/AHS 28 Structures, Structural Dynamics and Materials Conference, Monterey, CA, April 6-8, 1987.
7. Rogers, James L.; Young, Katherine C., and Barthelemy, Jean-Francois M.: "Distributed Computer System Enhances Productivity for SRB Joint Optimization", NASA TM 89108, February 1987.
8. Rogers, J. J., Jr.; Sobieszczanski-Sobieski, J.; and Bhat, R. B.: "An Implementation of the Programming Structural Synthesis System (PROSSS)", NASA TM 83180, December 1981.
9. Padula, S. L., and Young, K. C.: "Simulator for Multilevel Optimization Research", NASA TM 87751, June 1986.

APPENDIX

This appendix contains the DCL procedure files used to implement the distributed system for the multilevel simulator described in this paper. Figure 3 is a schematic of the 25 variable test case and figure 4 is a mapping of this test case onto five MicroVAX computers.
Each subsystem is initialized with the following files:

LEViSj.IN	input data (constant throughout test case)
LEViSj.EXE	FORTRAN optimization code
RUNij	DCL command file
V.DAT	input parameters

Each subsystem communicates with the data manager using the following files:

LEViSj.CM	input data from data manager
LVij.CM	output data sent to data manager where i denotes level number

and j denotes which subsystem on that level
The DCL commands are of the five types defined below. Although a choice of network option is used in the test case, only DECnet commands are included here for simplicity.

TYPE I Interactive procedure for total system

```
$! PROCEDURE TO DISTRIBUTE JOBS AND DATA TO WORKSTATIONS
$!          WRITTEN IN DEC COMMAND LANGUAGE (DCL)
$!
$ ON ERROR THEN GOTO ERR
$!
$! INITIALIZE AND INQUIRE ABOUT INPUT PARAMETERS
$ INQUIRE TIMEL " ENTER TIME LIMIT FOR THE PROCESSES IN
MINUTES"
$!
$! LEVEL AND subsystem INFORMATION
$!
$ INQUIRE NL "PLEASE ENTER NUMBER OF LEVELS"
$ INQUIRE NUML3 "ENTER NUMBER OF SUBSYSTEMS IN BOTTOM
LEVEL"
$ INQUIRE NUML2 "ENTER NUMBER OF SUBSYSTEMS IN NEXT LEVEL"
$ INQUIRE NUML1 "ENTER NUMBER OF SUBSYSTEMS IN TOP LEVEL"
$!
$! DETERMINE MAXIMUM NUMBER OF SUBSYSTEMS ON A LEVEL
$!
$ MAXS = NUML1
$ IF NUML1 .LT. NUML2 THEN MAXS = NUML2
$ IF NUML3 .GT. MAXS THEN MAXS = NUML3
$!
$! SELECT WORKSTATIONS
$!
$ INQUIRE N1 "ENTER WORKSTATION NAME FOR THE DATA
MANAGERS"
$ NCT = 1
$ REPEAT:
$ NCT = NCT + 1
$ INQUIRE N'NCT' "PLEASE ENTER WORKSTATION FOR SUBSYSTEMS
"
$ IF NCT .LE. MAXS THEN GOTO REPEAT
$!
$!  CLEAN UP FILES ON WORKSTATIONS
$!
$ DEL [SIM]*.OUT;*
$ DEL [SIM...]*.OUT;*
$ DEL [SIM..]*.CM;*
$!
$!  COPY DATA FILES AND DATA MANAGERS TO WORKSTATION
$!
$ COPY V.DAT [SIM.LEVEL3]V.DAT
$ COPY DATA_MANAGER.EXE [SIM.LEVEL3]*
$ COPY DATA_MANAGER.EXE [SIM.LEVEL2]*
$ COPY TOPLEV.EXE [SIM.LEVEL1]*
$ COPY RUNC23.COM [SIM.LEVEL3]RUNC23.COM
$ COPY RUNC12.COM [SIM.LEVEL2]RUNC12.COM
$ COPY RUNT.COM [SIM.LEVEL1]RUNT.COM
$!
$! COPY FILES FOR SUBSYSTEMS TO EACH LEVEL OF WORKSTATION
$!
$ LEVEL1 = 0
$ LEVELCT = 0
$ LEVELCYCLE:
$ LEVELCT = LEVELCT + 1
$ IF (LEVELCT.GT.NL) THEN GOTO CONTINUE
$ SUBLIM = NUML'LCT'
$ SUBSYSTEM_COUNT = 0
$ CT = 2
$ LEVEL1 = LEVEL1 + 1
$ SUBSYSTEMCYCLE:
$ SUBSYSTEM_COUNT = SYBSYSTEM_COUNT + 1
$ IF (SUBSYSTEM_COUNT.GT.SUBLIM) THEN GOTO LEVELCYCLE
$ WORKSTATION = N'CT'
$ DEL 'WORKSTATION'::[SIM.LEVEL'LEVEL1']V.DAT;*
$ DEL 'WORKSTATION'::[SIM.LEVEL'LEVEL1']*.CM;*
$ FILE1 = F$FAO("!SL!SL",LEVEL1,SUBSYSTEM_COUNT)
$ FILE2 = F$FAO("!AS!SL!SL!AS","RUN",LEVEL1,subsystem_count,".com")
$ FILE3 =
F$FAO("!AS!SL!AS!SL!AS","LEV",LEVEL1,"S",subsystem_count,".EXE")
$ FILE4 =
F$FAO("!AS!SL!AS!SL!AS","LEV",LEVEL1,"S",subsystem_count,".IN")
$!   USE DECNET
$!
$ COPY1:
$ COPY 'FILE2' 'WORKSTATION'::[SIM.LEVEL'LEVEL1']*
$ COPY 'FILE3' 'WORKSTATION'::[SIM.LEVEL'LEVEL1']*
$ COPY 'FILE4' 'WORKSTATION'::[SIM.LEVEL'LEVEL1']*
$ IF LEVEL1 .NE. NL THEN GOTO NEXT
$ COPY V.DAT 'WORKSTATION'::[SIM.LEVEL'LEVEL1']*
$ NEXT:
$!
$!   SUBMIT COMMAND FILE TO BATCH QUEUE
$!
$ SUBMIT/REMOTE 'WORKSTATION'::[SIM.LEVEL'LEVEL1']RUN'FILE1'
-/PARAMETERS=('N1')
$ CT = CT + 1
$ GOTO subsystemCYCLE
$!
$!    SUBMIT DATA MANAGER COMMAND FILES TO BATCH QUEUE
$!
$ CONTINUE:
$!
$ IF NL .EQ. 2 THEN GOTO LEVEL2
$ SUBMIT/NOPRINT [SIM.LEVEL3]RUNC23
-  /PARAMETERS=('NUML3','NUML2','N2','N3','N4','N5',)
$ LEVEL2:
$ SUBMIT/NOPRINT [SIM.LEVEL2]RUNC12 -
/PARAMETERS=('NUML2','NUML1','N2','N3','N4','N5',)
$ SUBMIT/NOPRINT [SIM.LEVEL1]RUNT -
/PARAMETERS=('NUML1','NUML3','NL','N2','N3','N4','N5')
$ SUBMIT/NOPRINT [SIM.LEVEL1]TIMECK -
/PARAMETERS=('TIMEL','N2','N3','N4','N5')
$ GOTO FINISH

$ ERR:
$ WRITE "YOU HAVE AN ERROR IN THIS PROCEDURE"
$ FINISH:
```

TYPE II Batch file for data managers

```
$!
$!       RUNC12.COM -- DATA MANAGER BETWEEN LEVELS 1 AND 2
$!
$!   INPUT
$!       P1 = # OF SUBSYSTEMS IN LEVEL 2
$!       P2 = # OF SUBSYSTEMS IN LEVEL 1
$!       P3 = NODE NAME FOR LEVEL1S1 RUN
$!       P4 = NODE NAME FOR LEVEL1S1 RUN
$!       P5 = NODE NAME FOR LEVEL1S3 RUN
$!
$!    CLEAN UP FILES AND INITIALIZE
$!
$ SET DEF [SIM.LEVEL2]
$ DEL OUT.DAT;*
$ PURGE *.*
$ SET VER
$ START1:
$ VNUM = 2
$ LSUB = P1
$ NSUB = 0
$ FNUM = 20
$!
$!     BEGIN WAITING LOOP FOR INPUT DATA FILES
$!
$ BEGIN:
$ IF LSUB .EQ. 0 THEN GOTO CHECKV
$ LSUB = LSUB - 1
$ FNUM = FNUM + 1
$!
$!     CONSTRUCT INPUT FILE NAMES
$!
$ UNIT = F$FAO("!AS!SL!AS","LV",FNUM,".CM")
$ START:
$ SET MESSAGE/NOF/NOS/NOI/NOT
$ NOTOPEN1:
$!
$!   CHECK FOR TERMINATION  FILE OUT.DAT AND INPUT DATA FILES
$!
$ FILEX = F$ F$SEARCH("OUT.DAT")
$ IF FILEX .NES. "" THEN GOTO FINISH
$ WAIT 00:00:05.00
$ OPEN/READ/ERROR=NOTOPEN1 FILE 'UNIT'
$ CLOSE FILE
$ GOTO BEGIN
$!
$!   CHECK FOR INPUT FILE V.DAT
$!
$ CHECKV:
$ FILEX = F$ SEARCH("OUT.DAT")
$ IF FILEX .NES. "" THEN GOTO FINISH
$ WAIT 00:00:05.00
$ OPEN/READ/ERROR=CHECKV FILE V.DAT
$ CLOSE FILE
$ SET MESSAGE/F/S/I/T
$!
$!    RUN DATA MANAGER PROGRAM
$!
$ RUN DATA_MANAGER
$!
$!    SEND COMBINED DATA UP TO SUBSYSTEMS ON NEXT LEVEL
$!
$ CT = 3
$ SEND_DATA:
$ WORKSTATION =P'CT'
$ NSUB = NSUB + 1
$ LVS = F$FAO("!AS!SL!AS","LEVEL1S",NSUB,".CM")
$ COPY  'LVS' 'WORKSTATION'::[SIM.LEVEL1]'LVS'
$ COPY  V.DAT 'WORKSTATION'::[SIM.LEVEL1]V.DAT
$ IF NSUB .EQ. 'P2' THEN GOTO START4
$ CT = CT + 1
$ GOTO SEND_DATA
$ START4:
$ COPY VS.DAT [SIM.LEVEL1]V.DAT
$ GOTO START1
$ FINISH:
```

TYPE III   Batch Procedure for each subsystem

```
$!
$!          PROCEDURE RUN11 EXECUTES LEVEL1S1
$!
$!       INPUT
$!         P1 = WORKSTATION NAME WHERE TOPLEV RUNS
$!
$!
$ SET DEF [SIM.LEVEL1]
$ DEL *.OUT;*
$ DEL OUT.DAT;*
$ PURGE *.*
$ SET VER
$ START:
$ SET MESSAGE/NOF/NOS/NOI/NOT
$ NOTOPEN:
$!
$!    CHECK FOR TERMINATION FILE OUT.DAT
$!
$ FILEX = F$SEARCH("OUT.DAT")
$ IF FILEX .NES. "" THEN GOTO FINISH
$ WAIT 00:00:05.00
$ RUNIT:
$!
$!    WAIT FOR INPUT DATA FILES V.DAT AND LEVEL1S1.CM
$!
$ OPEN/READ/ERROR=NOTOPEN FILE V.DAT
$ CLOSE FILE
$ SET MESSAGE/F/S/I/T
$ NOTOPEN2:
$ FILEX =F$SEARCH("OUT.DAT")
$ IF FILEX .NES. "" THEN GOTO FINISH
$ WAIT 00:03:05.00
$ RUNIT:
$ OPEN/READ/ERROR=NOTOPEN2 FILE LEVEL1S1.CM
$ CLOSE FILE
$ RUN2:
$!
$!    RUN LEVEL 1 SUBSYSTEM 1 OPTIMIZATION CODE
$!
$ RUN LEVEL1S1
$!
$!    USE DECNET TO COPY LV11.CM TO LEVEL DATA MANAGER
$!
$ COPY [SIM.LEVEL1]LV11.CM  'P1'::[SIM.LEVEL1]*
$ GOTO START FINISH:
```

TYPE IV   Batch file to manage data for top level and run system optimization

```
$!       PROCEDURE RUNT.COM  RUNS TOPLEVEL OPTIMIZATION
$!
$!
$!        INPUT
$!          P1  NUM OF SUBSYSTEMS IN LEVEL 1
$!          P2  NUM OF SUBSYSTEM IN BOTTOM LEVEL
$!          P3  NUM OF LEVELS
$!          P4  WORKSTATION NAME FOR BOTTOM LEVEL SUB 1
$!          P5  WORKSTATION NAME FOR BOTTOM LEVEL SUB 2
$!          P6  WORKSTATION NAME FOR BOTTOM LEVEL SUB 3
$!          P7  WORKSTATION NAME FOR BOTTOM LEVEL SUB 4
$!
$ ON ERROR THEN GOTO ERR
$ SET DEF [SIM.LEVEL1]
$ DEL OUT.DAT;*
$ PURGE *.*
$ NUM_SYSB = P2
$ NUM_SYS1 = P1
$ SET VER
$ START:
$ NB1 = 1
$ NBB = 0
$ SET MESSAGE/NOF/NOS/NOI/NOT
$ NOTOPEN:
$!
$!    CHECK FOR TERMINATION FILE OUT.DAT
$!
$ FILEX = F$SEARCH("OUT.DAT")
$ IF FILEX .NES. "" THEN GOTO STOP
$ WAIT 00:00:05.00
$!
$!    WAIT FOR INPUT FILE FROM LEVEL 1 SUBSYSTEMS
$!
$ IF(NB1.GT.NUM_SYS1) THEN GOTO NEXT
$ OPEN/READ/ERROR=NOTOPEN FILE LV1'NB1'.CM
$ CLOSE FILE
$ NB1 = NB1 + 1
$ GOTO NOTOPEN
$ NEXT:
$ OPEN/READ/ERROR=NOTOPEN FILE V.DAT
$ CLOSE FILE
$!
$!   RUN TOP LEVEL OPTIMIZATION CODE
$!
$!
$ SET MESSAGE/F/S/I/T
$ RUN TOPLEV
$!
$!   CHECK FOR TERMINATION FILE  OUT.DAT
$!
$ FILEX = F$SEARCH("OUT.DAT")
$ IF FILEX .NES. "" THEN GOTO STOP
$!
$!   SEND INPUT FILES TO BOTTOM LEVEL SUBSYSTEMS
$!
$ CT = 4
$ REPEAT:
$ WORKSTATION = P'CT'
$ NBB = NBB + 1
$ COPY LEV'P3'S'NBB'.CM
'WORKSTATION'::[SIM.LEVEL'P3']LEV'P3'S'NBB'.CM
$ COPY VS.DAT 'WORKSTATION'::[SIM.LEVEL'P3']V.DAT
$ CT = CT + 1
$ IF NBB .LT. NUMSYSB THEN GOTO REPEAT
$ COPY  VS.DAT [KCY.LEVEL'P3']V.DAT
$ PURGE *.SAV
$ GOTO START
$ STOP:
$ DONE:
$!
$!   COPY OUTPUT FILE TO PRINTER
$!
$ PRINT TOPLEV.OUT
$ GOTO FINISH
$ ERR:
$ COPY [SIM]ERR.DAT OUT.DAT
$ FINISH:
```

Type V Batch Procedure that acts as time checker and system terminator

This procedure file must be in the same directory as the TOPLEVEL procedure file

```
$!
$!              PROCEDURE TIMECK.COM
$!
$!        INPUT
$!            P1 = TIME LIMIT IN SECONDS
$!            P2 = NODE NAME FOR OTHER PROCESSOR
$!            P3 = NODE NAME FOR OTHER PROCESSOR
$!            P4 = NODE NAME FOR OTHER PROCESSOR
$!            P5 = NODE NAME FOR OTHER PROCESSOR
$!
$ ON ERROR THEN GOTO STOP
$!
$ SET VER
$ TIME = F$TIME ( )
$ TIMEL = P1
$ TIMEC = 0
$ SET DEF [SIM.LEVEL1]
$!
$!     CHECK TIME
$!
$ START:
$ TIMEC = TIMEC + 10
$ IF TIMEC .GT. TIMEL THEN GOTO STOP
$!
$! CHECK FOR TERMINATION FILE GENERATED BY TOPLEV
$!
$ FILEX = F$SEARCH("OUT.DAT")
$ IF FILEX .NES. "" THEN GOTO FINISH
$ CONTINUE:
$ WAIT 00:00:10.00
$ GOTO START
$ TIME= F$TIME( )
$ SHOW SYMBOL TIME
$ STOP:
$!
$! SEND TERMINATION FILE  OUT.DAT TO STOP ALL PROCESSORS
$!
$ COPY OUT.ERR OUT.DAT
$ FINISH:
$ COPY OUT.DAT WORKSTATION'P2':: [SIM.LEVEL1]OUT.DAT
$ COPY OUT.DAT WORKSTATION'P5':: [SIM.LEVEL1]OUT.DAT
$ COPY OUT.DAT WORKSTATION'P3':: [SIM.LEVEL1]OUT.DAT
$ COPY OUT.DAT WORKSTATION'P2':: [SIM.LEVEL2]OUT.DAT
$ COPY OUT.DAT WORKSTATION'P5':: [SIM.LEVEL2]OUT.DAT
$ COPY OUT.DAT WORKSTATION'P3':: [SIM.LEVEL2]OUT.DAT
$ COPY OUT.DAT WORKSTATION'P4':: [SIM.LEVEL2]OUT.DAT
$ COPY OUT.DAT [SIM.LEVEL2]*
$ COPY OUT.DAT WORKSTATION'P3':: [SIM.LEVEL3]OUT.DAT
$ COPY OUT.DAT [SIM.LEVEL3]*
$MAIL/SUBJECT="PROCESSES SHUTDOWN" OUT.DAT  SIM
$ EXIT:
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-101519	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle A Strategy for Reducing Turnaround Time in Design Optimization Using a Distributed Computer System	5. Report Date October 1988
	6. Performing Organization Code
7. Author(s) Katherine C. Young, Sharon L. Padula, and James L. Rogers	8. Performing Organization Report No.
	10. Work Unit No. 505-63-01-07
9. Performing Organization Name and Address NASA Langley Research Center, Hampton, VA 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

Presented at The 1988 ASME Design Technology Conferences - The Design Automation Conference, Kissimmee, Florida, September 25-28, 1988

16. Abstract

There is a need to explore methods for reducing lengthly computer turnround or clock time associated with engineering design problems. Different strategies can be employed to reduce this turnaround time. One strategy is to run validated analysis software on a network of existing smaller computers so that portions of the computation can be done in parallel. This paper focuses on the implementation of this method using two types of problems. The first type is a traditional structural design optimization problem, which is characterized by a simple data flow and a complicated analysis. The second types of problem uses an existing compter program designed to study multilevel optimization techniques. This problem is characterized by complicated data flow and a simple analysis. The paper shows that distributed computing can be a viable means for reducing computational turnaround time for engineering design problems that lend themselves to decomposition. Parallel computing can be accomplished with a minimal cost in terms of hardware and software.

17. Key Words (Suggested by Author(s)) Parallel computing, Distributed processing, Multilevel optimization	18. Distribution Statement Unclassified - Unlimited, Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	9	A02

NASA FORM 1626 OCT 86