Assignment

REALTIME FAULT TOLERANT SYSTEM

Anila Joseph

Student ID:15657697

THIS PAGE IS NOT USED

TABLE OF CONTENT

Topic Page

1) INTRODUCTION.. 1

2) DESIGN GOALS OF THE TANDEM 16 3

3) SYSTEM ARCHITECTURE OF THE TANDEM 16.. 5

4) FAULT TOLERANCE STRATEGIES INVOLVED IN THE TANDEM 16 9

5) THE TANDEM 16 SOFTWARE RELIABILITY... 13

6) APPLICATION 15

7) CONCLUSION 21

THIS PAGE IS NOT USED

INTRODUCTION

The rapid growth in technology has increased the demand for online processing in many businesses as we notice many organisations have switched over from manual and batch methods to online computer processing because of which it has become increasingly vulnerable to computer failures. As observed the failures of an online system results in immediate business losses when compared to batch system, the direct costs of failure might simply be increased overtime for the operating staff. The Tandem 16 (1,2) ease significantly designed to provide more reliability than current available commercial computer system. The Tandem 16 (1,2) was designed to provide a system for online application. The hardware structure comprises of multi process modules which are interconnected by redundant interprocessor buses each processor has its own memory, power supply, I/O channel which are in turn connected to all other processors by redundant interprocessor buses. Since each I/O controller is redundantly powered and is connected to two different I/O channels. Therefore the failure of any interprocessor bus will not affect the ability of the processor to communicate with other processor; similarly the failure of an I/O channel or a processor will not cause the loss of an I/O device. Likewise the failure of a module does not disable any other module or inter-module communication. Finally certain I/O devices such as disc devices are connected to two different I/O controllers and these disc drivers are in turn duplicated such that the failure of one I/O controller or disc drive does not effect the data. The system is not a live multiprocessor but rather a multiple computer whose approach is preferable for several reasons.

The advantages are as follows: since no module is shared by the entire system it increase the system reliability. Secondly a multiple computer system does not require complex hardware to handle multiple access paths to a common memory. In smaller systems the cost of such multiported memory is undesirable compared to larger system performances suffer because of memory access interferences online repair is essential as reliability. The module structure of Tandem 16 system allows processors, I/O controllers, buses to be repaired while the rest of the system continues to operate once repaired they can be reintegrated into the system. The system structure supports a wide range of system sizes. As many as sixteen processors each with upto 512k bytes of memory are connected into one system. Each processor can also have upto 256 I/O devices that are to be connected to it. This provides tremendous growth of application programs and also processing loads without the requirement that the application be reimplement on a larger system with a different architecture. as a result the system is suppose to provide a general solution to the problem of providing a failure tolerant, online environment which is suitable for commercial use, the system supports conventional programming languages and peripherals, it also provides a large number of terminals with access to large data buses

THIS PAGE IS NOT USED

DESIGN GOALS OF THE TANDEM 16

The Tandem 16 system was designed to solve a problem which was not stated in terms of the hardware and software required but in terms of system requirements. The design of the hardware and software then proceeded in tandem to produce a unified solution. The hardware design was concerned with the contents of each module, the interconnection to common buses, error detection and correction within modules and communication paths. The software design was given the responsibility of selecting the right modules that are to be used and which buses are to be used to communicate with them it was also given that the responsibility to control recovery actions whenever errors were detected.

OPERATING SYSTEM DESIGN GOALS

The most important goal of the operating system, Guardian was to provide a failure tolerant system. This resulted into the following design "axioms":

1. Even after a single detected module or bus failure the operating system should be able to remain operational
2. The operating system should permit any bus or module to be repaired online and also integrated into the system
3. The operating system must be implemented in a reliable and effective manner. the increase in the reliability provided by the hardware architecture must not be negated by any software problems with the great sizes and number of hardware configuration that are possible giving rise to the second set of requirements
4. The operating system should be able to support all possible hardware configurations ranging from two processor, the disc less system through a sixteen-processor system consist of billions of bytes of disc storage
5. The operating system must hide the physical configuration such that applications could be written and run on a great variety of system configurations

THIS PAGE IS NOT USED

SYSTEM ARCHITECTURE OF THE TANDEM 16

Operating system architecture

To implement all the above capabilities the OS was designed as a multiprocessor on Dijikstras work

The operating system consists of the following modules:

1. Processes:

   Many processes can be there inside one processor. Every process is generally unique to the processor in which it is created and may not work in another processor. Execution priority is assigned to every process via a process time which is from the highest priority ready process to the lowest one. Process local event and counting semaphores are the primitives for process synchronization. PSEM and VSEM functions are used based on the p and v functions in Dijikstras work when a semaphore operation is carried out. Semaphores are used for synchronization only when both the processes belong to the same processor. Some of the tasks they carry out are like control access to message control blocks. An event flag is raised as a result of a low level action like processor power on. A process might wait for a couple of events to occur with the help of function WAIT. On occurrence of the first waited for the process is activated. AWAKE function is put to use for signalling of events. To prevent the loss of an event when a process is signalled and an event just in case is not waiting for it, the wake up waiting mechanism is used to queue up the signals. The event flags are defined beforehand for upto 8 processors and like semaphores event signals also work for the processes within the same processor. A maximum block time is allotted to the process when it wants itself to be blocked for a particular event or semaphore. Once this time limit expires without the process occurring then normal execution it is resumed but after returning an error condition. With the help of this timeout watch dog timers can placed on device interrupts where a failure might occur.

2. Messages:

   A message is sent to the desired server process by the procedure LINK for a process request for service. The message consists of two parameters namely the type of request and data needed. An event flag is set once message is queued for server process and then the requestor processor continues on. When server processor wants to check for any messages it calls LISTEN. If messages are there then LISTEN returns the first message that is queued otherwise it indicates that no messages are queued.

Then the procedure READLINK is used to obtain a copy of the requestor's data and then server process processes the request. Any information regarding the result of process is given to the requestor process by the WRITELINK procedure through a signal by the event flag. Transaction is completed on the requestor side via the BREAKLINK procedure.

The system is designed in such a way that an error will only occur if the sender and receiver processes fail or their processors fail. The design also consists of communication protocol for the interprocessor buses so that they can handle upto a single error during execution of a message and these errors are corrected as the message is continuing on and put back as well. The interprocessor buses are not meant for the processes within the same processor and are done faster in the memory and the processes involved in message transfer can't see this difference. The resources needed for message transmission are provided to the message transfer request at the start. Once the link is complete both processes assured that enough resources are available. To prevent deadlocks in message transmission control blocks are reserved by the processes and that it continues to maintain a hassle free communication between sender and receiver.

3. Process Pairs:

This mechanism provides a system wide access to the I/O devices. Two cooperating processes located at two different processors and controlling one I/O device make up an I/O processor pair. One process is called the primary and the other is called the backup. Controlling the I/O device and handling requests is upto the primary process. Like in the case of a file open process when a request is sent the backup system gets to know about the message system because of the primary process. This is done in case a primary process fails or the I/O channel error occurs, then the backup process can take over and continue with the operation. A block of driver code can't be used in the system to call on a device due to the distributed nature of the system. This feature of the Tandem 16 is either known by a logical device name or by a logical device number. The logical device table maps the device name to the processes in each processor. It also supplies to process ID's to each device. Message requests are made with respect to the device name or number and accordingly a message is sent to the first process in table. An error indication returns if a message sent to backup process or not sent at all. To resend the message the process ID's are reversed. Here error recovery is done automatically. The requestor is not concerned with the process while handling the request.

A fault tolerant method is provided for interprocess communication by the two primitive's processes and messages. Processor pairs provide us with a uniform access to an I/O device. It works irrespective of things like process implementations. This method makes use of checkpoints so that backup process can takeover in case of failure. This checkpoint process works independently on its own.

4. System Processes:

In this step functions are assigned to processes and another process pair called the operator comes into picture. Printing and formatting of error messages in the system console are upto this pair. This process has the capability to reserve message control blocks and sees that no blockage is caused due to resource problems.

We can explain with the example of scanning, in this case requests are sent to a terminal to scan and the terminal is more intent on sending the error messages to the process rather than completing the request first, this would create an infinite loop that never ends and to prevent this the error messages are terminated beforehand in this system. Each processor has a system monitor process which carries out functions like process deletion. The processor also has a memory management process in it which allocates a page of physical memory and sends the required messages to disc processes for actual disc I/O and the pages are brought in on a demand basis. The system makes use of not very sophisticatedly design algorithms.

5. Application Process Interface:

This provides us with a library of procedures to access system resources. They run in a calling process environment and may or may not send messages to other processes. When we use the time procedure, it just returns the current time and doesn't send any messages. Here it is seen that resources of system easily accessed by procedure calls and the process structure and system remain hidden.

6. Initialization and Processor Reload:

A system is initialized when one processor is cold loaded from some disc on system. The load file consists of a memory image of the OS, resident code and data along with all processes in existence in their respective initial states. Then a command interpreter process is created by the system monitor process. Gaurdian can be brought back even when a processor or peripheral devices are not functioning. This is possible due to three basic factors

i) disc images of the OS are kept on multiple disc drives,
ii) access of the I/O controller is not dependent on only one processor and
iii) the switch register of the processor selects the terminal that has the initial command interpreter on it.

The system consists of one processor and any peripherals after a cold load. Both numbers can be increased by the following command interpreter command i.e. RELOAD 1, SDISC. This command reads the disc image for processor 1 from disc SDISC and sends it to the interprocessor bus of processor 1. Once it is loaded all remaining processors are notified that processor 1 is in operation.

Another function of this very command is to reload the processor after repair. The Gaurdian can't differentiate between an initial and a later load. In both cases resources are added so that they can be fully utilized. The reload message operation hence clearly depicts how two functions are split in the Gaurdian.

7. Operating System Error Detection:

Apart from the various error correction and detection mentioned before some more software error checks are also provided. One of them is the detection of a processor which is down. The way this thing works is that every second each processor sends a message 'I AM ALIVE' and every two seconds it is checked by the other processors and when the other processor doesn't receive a message it assumes that the processor is dead.

Some checks are carried out by the operating system on the data structures used and if a processor finds such an error it comes to a halt. Watch dog timers are used by the I/O interrupts so that the system doesn't hang up if an I/O operation is not completed as per the expected interrupt. The backup process takes over with the second I/O bus when an I/O error occurs. Additional checks are carried out on the control information received over the buses along with the receiver processor state.

FAULT TOLERANCE STRATEGIES INVOLVED IN THE TANDEM 16

1. APPLICATION PROGRAMS CHECKPOINTING

Services are provided to a failure tolerant by the application process pairs results are returned to the requestor process based on the request processed by the primary process in any case if the primary process fails the backup must be able to continue the service for this any changes performed on the state of the primary process must be sent to the backup process while such checkpoints could be sent by instruction by instruction basis this is not feasible because of the overhead involved. checkpoints are needed only when the state is about to make a non-retryable request to another process for example, when a primary process and its backup are in the same state the primary processes starts some operation at time T1 later it will checkpoint the changes made since time T1 to its backup when it is ready to write the result to a disc file at time T2 the process is again in the same state the backup process would restart at the last checkpoint made at T2 if the primary process failed at any point before T2 to defining the restart points for jods in the batch processing system the selection of states to checkpoints is analogous in batch environments the checkpoints are saved in a disc file whereas in process-pairs they are saved in a backup process system functions are provided by guardians for checkpointing process state information between processes of a process-pair the portions of the process data space is the first type of checkpointed which includes global data, the current stack, holding procedure return addresses, procedure local variables, procedure parameters. Considering the following program segment which is written in T/TAL whose purpose is to output the first 200 items of an array, “buffer”:

```
FOR j: = 1 TO 200 DO
BEGIN
CALL WRITE (terminal,buffer[j],itemlen);
END;
```

Two calls to the checkpoint procedure could make the operation failure tolerant the entire buffer is copied to the backup process in the first checkpoint this is only done once the data is not changed by later processing. Before each write the current process state is saves including the variable j for the second checkpoint, allowing the backup process to take over the operation duplicating at most one line of the output.

```
CALL CHECKPOINT (buffer[1],buffersize);
FOR j: = 1 TO 200 DO
BEGIN
CALL CHECKPOINT (stackbase);
CALL WRITE (outfile,buffer[i],itemlen);
END;
```

The backup takes over at the last checkpoint whenever the primary process fails. The process were to copy the next two hundred values to be the output from some disc file this would be the next logical segment

```
FOR j: = 1 TO 200 DO
BEGIN
CALL READ (infile,buffer,itemlen);
CALL WRITE (outfile,buffer.itemlen);
END
```

SEQUENCE PRIMARY	VALUES BACK UP	AFTER ACTION DISC PROCESS
0	0	0
1	0	1
-	1	1

From the case mentioned above not only would the process data space contents need to be checkpointed but also the current file pointers for the input and output files, making sure that they are properly set when the backup process takes over both processes of the process-pair must have the file opened in order for the file pointer to be checkpointed special functions are provided to the primary process to open and close a file by the backup process

```
CALL CHECKOPEN (filename,….)
CALL CHECKCLOSE (filename,…)
```

In the example mentioned above, when the primary process starts CHECKOPEN would be called followed by the call to open the following is the view of the program segment

```
CALL CHECKPOINT (, buffer[1],buffersize);
FOR j :=1 TO 200 DO
BEGIN
CALL CHECKPOINT (stackbase, ,infile, ,outfile);
CALL READ (infile,buffer,itemlen);
CALL WRITE (outfile,buffer,itemlen);
END;
```

The backup would take over and repeat the read using the same file pointer as was used by the primary, if a failure has to occur after the read but before the write in the above example the result in the record could be written twice if a failure following the write but preceding the next checkpoint, if the record is being written to the absolute same position in the file it would cause no problem when writing to the key sequence disc file however an error might occur in such cases, the primary would successfully write the record to the file but when repeating the write the backup would get a duplicate key error by having guardian automatically generate an optional sequence number for the disc files writes this problem could be solved

When a file is checkpointed it is the sequence number for the next write file a part of the information will be copied to the backup process the sequence number passed by the file system will be compared with the copy that is held by the disc process this is possible only when a write is done to a file if the result is the same the operation is done and the result is returned to the application process and a copy is saved by the disc process the copy of the previous operation status will be returned and no operation will be performed if the sequence number does not agree.

Any records locked in the files will be considered locked by the process pair if the process pair has a file open if the primary fails the backup will finish the file modification with the locks still in effect to preserve the integrity of the data base. The backup process will receive the checkpoint information through the call to the procedure CHECKPOINTOR this is done while the primary process is operating the procedure moves the checkpointed portion of the primary process data space into the backup data space and it will save the latest file information this is performed only when the primary process sends a checkpoint message through a call to the checkpoint if the primary process exists when a message is directed to the backup process it will be rejected with a ownership error this will inform the sender that the message is supposed to be sent to the other member of the process pair whenever the primary process fails the checkmonitor will transfer the control to the correct restart point.

A similar checkpoint facility is provided by the tandem implementation of Cobol .in this case the checkpoint is not a automatic operation a checkpoint scheme that could be analysed for correctness is yield only when the application design phase results in fewer checkpoints attention should also be given as to how an application will recover from failures that are occurring while a write operation is in progress to non disc devices by rewriting the entire screen a recovery while accessing a CRT terminal could be automatically done some manual intervention and operation interaction with the application program this is required when recovery of printing checks on a line printer is required

2. APPLICATION STRUCTURING

General tools for application structuring are provided by the process, process-pair, Interprocess communication primitives of guardians For example consider a inquiry application such as cinema reservation requests comes in from different types of people for reservations, cancellations and cinema registration. Other requests come in for items such as the management reports that include the availability of shows, tickets, seats for other cinema on the same date the application could be structured as follows the process pairs are defined for different types of terminals to handle the actual terminal I/O and also the initial request verification every process pair is designed to handle different terminals whenever a valid request is received from a terminal the process pairs will route that message to the appropriate server process-pair each server process-pair is assigned with a certain part of the application in some cases multiple copies of the server program is run to allow multiple request to be processed in parallel the advantages to this approach are firstly it has cleanly separated the handling of terminals and processing of request by adding a type of terminal control process-pair a new terminal is added by adding another type of server process-pair a new type of request can be handled similar software modification and testing can be done on modular basis however in this given structure there is no requirement for a specific number of processors in the system or the relative location of processes without disturbing the application's internal structure the number of processors, the amount of memory and the physical location of the processes can be changed as the system load or the application changes

THE TANDEM 16 SOFTWARE RELIABILITY

The archetypal system crash was hoped to be eliminated much as possible when the operating system design started the system crashes in a non repeatable fashion one or twice a day to show that we have achieved that goal the experience on a in-house system was used primarily for software development and manual writing a processor failed because of a software problem during the three-month period in summer of 1977 on two occasions in both the cases the problem was found and the failure was repeated by running that particular program to propose the explanation of this reliability much time was spend in initially specifying primitives and the system was structured carefully as they gained experience in applying these primitives at a higher level they found problems resulting in changes in the lower level than "Kludges" in the higher level the design structure was forced to stay within the implementation because of the distribution nature of the hardware if by using processes interaction through messages the problem could not be solved then it could not be "Kludged" by turning off interrupts or by changing some flags in the memory there is a very temptation to solve difficult problems in this manner given a single processor system

Secondly as the hardware and the operating system was developed parallel there was another vendor system which was used to provide interactive text editing , a cross T/TAL compiler, a Tandem/16 processor simulator, a downloader for Tandem 16 prototypes as each level of the system was implemented the implementation and the checkout were not impeded by unreliable prototypes and this could be extensively checked throught the level of command interpreter the tools would provide initial implementation and checkout to all functions of the system the knowledge of this approach can be best shown as a fact that the first prototype processor was made available to the operating system group, all the operating system function that ran on the simulator also ran on the prototypes

Thirdly from the start the debugging tools was built into the operating system breakpoint were allowed to be set at any level of the operating system including the interrupt handler as a lower level interactive debugger was implemented once the low level debugger is entered into one of the processors the clocks in all the other processors in the system will be stopped so that they do not decide the first process that is down the rest of the system will continue if the first process continues to track the problems that manage that manage to destroy the low-level debugger a full maintenance panel should be used into the low level routine consistency checks are also coded considering an example before any element is inserted into a doubly-linked list the new elements that are being inserted behind are verified in a list links of elements in tracking problems these checks are proved to be extremely valuable or if new feature are to be implemented in the system the system has the property to stop at one of the consistency checks as soon as something is gone wrong which allows the problem to be rapidly found even when extensive changes are being made to the system

Fourthly at all the levels of the system formal testing is carried out as they were implemented a third person was added to the project well before it completed whose job was only testing to assure that all the system functions at all the levels were checked they tested not just the external specifications of the system but also the underlying system primitives

Finally making the primary design goal of the entire system reliable to control the implementation on daily basis the system design goals should be clearly mentioned and it should be understood by all who are involved whenever seemingly small decisions are made on the other hand the implied goals are forgotten

APPLICATION

1. Background:

   Raaga PVT LTD (RPL) is a private company which allows people to listen songs online. They have a huge collection of songs and videos. As the online users are increasing day by day, user's of this company are also increasing.

   The business of this company is as follows:

   1: Give the user a unique ID and password
   2: To check the account balance in the members account and then allow him to access the songs database.
   3: The users have to pay $2 for 10 mins for accessing the database

2. Problem of the company:

   The problem this company is facing is the server reliability. When the customer is accessing the site and if the server system goes down and the service is not provided, still the customer is charged. This makes many costumers unhappy.

3. How can Tandem 16 helps this company to solve there problem:

   By using Tandem 16 computer system, reliability of the computer increases and hence there problem can be solved.

DESIGN GOALS:

The design goals of using Tandem 16 for this application are as follows:

1. The Server system should be reliable:

   This means that the service should be provided continuously to the customer regardless of any failure within the system. For an example, if the user is using the data base and suddenly in between the disc drive of the server crash, inspite of this crash the system should give the correct output to the customer. Tandem 16 computer system provide this reliability for the 24/7 service providers.

2. The system should be capable of fault tolerant:

   Making the service fault tolerant by using the tandem 16 computer system here implies that each primary process in this application should have a redundancy i.e. duplication of all the S/W and H/W. That is it should duplicate the data as it is processed, so that failure at any point can restore the process which it was carrying.

3. The Software part and Hardware part of the system should be completely integrated with each other.

   This goal can be achieved by using the tandem 16 computer system. Hardware design is concerned with the contents of each module and there connection to the common buses. It was also concerned for error detection and correction within modules and on the communication path. Software is concern with control, which is selecting the module to use and the bus to use for communicating with them. More over software is responsible for recovery action when the error is detected.

<u>SYSTEM ARCHITECTURE:</u>

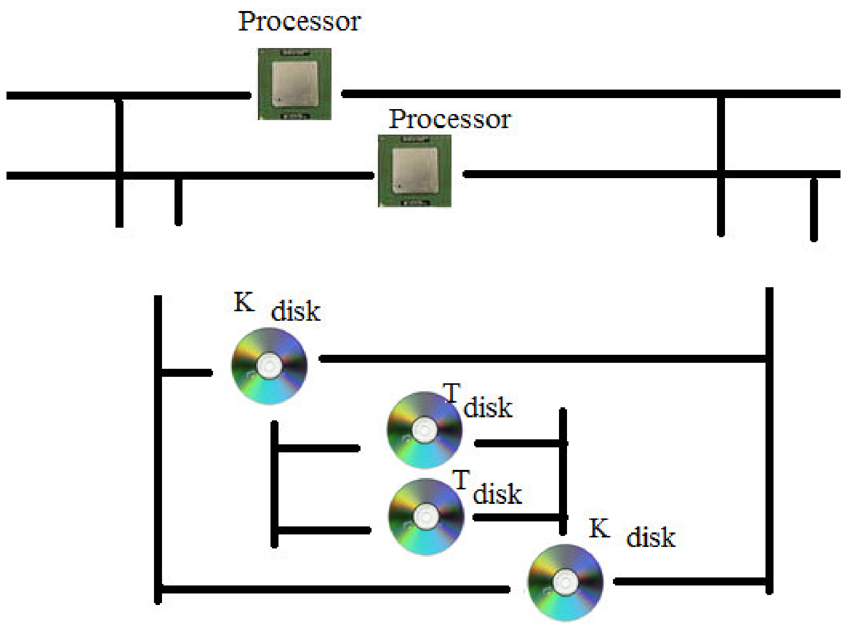


Fig 1

Explanation of the above model (Fig. 1)

Here the are 2 processor, 2 K disk and 2 T disk, which are all connected to each other in such a manner that failing of one module will not effect the other. If one of the modules goes down the other of same type goes into action state

Using Tandem 16 in the system architecture would improve the reliability and performance of Raaga PVT LTD server web service. The architecture of the Tandem computer system which includes messages, process, process pairs and error detection as its main goal would be of great use.

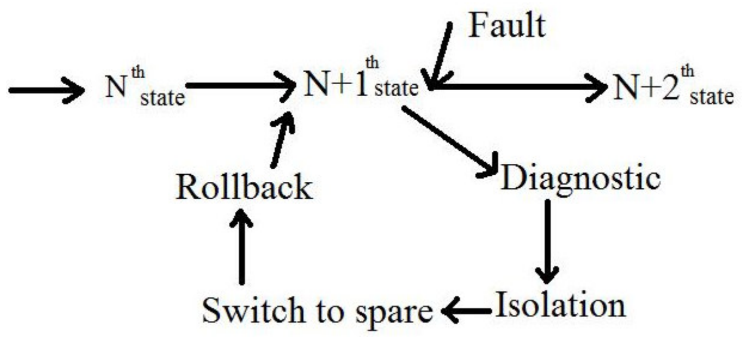


Fig 2

The above diagram show how the processor switches to other processor when the fault occurs. For an example processor is in Nth state then in N+1th state and then to N+2th state, while going from N+1th state to N+2 state say a fault have occur. Then the current processor is send to isolation and the spare processor is made active and rolled back to N+1 state (which was executed correctly by the pervious processor) and the previous processor is modified or replaced. Introducing this technique of failure tolerant in RPL server we can make it more reliable. Which is explained briefly later part of the application.

The RPL server also requires multiple processors because it needs to carry out multi tasks at a single given time .i.e. it needs to check the user ID and password, check the account , need to find out the balance of that particular customer and provide him with his songs and video.

Distinct processor and at some point handling multiple requests, hence there is a need of using various priority algorithm to sort different process. The process with the highest priority will execute before the process which have lower priority. Hence following the “THE” concept of semaphores is used for processes within the same processor. An event flag is raised, as the result of a low level action like device interrupts, processor on, message completion and message arrival.

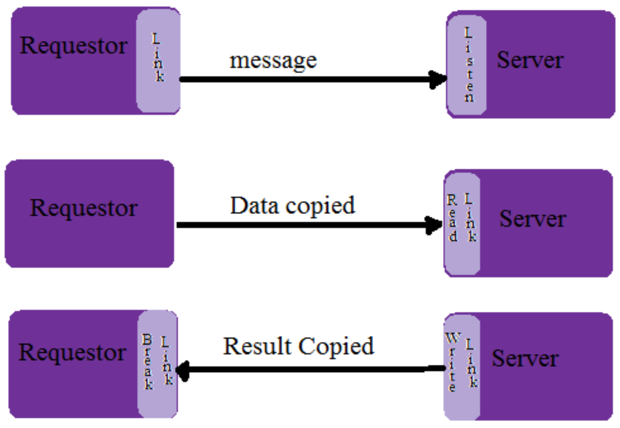
A process waits for a few events to occur when it sees the function WAIT. Ones the occurrence of the first waited for the process is activated AWAKE function is put on for use for signalling of events.

Event signal are queued by using a “wake up waiting” mechanism, so as if there is no lose of event when a process is not waiting on it processor stops itself and wait for a semaphore or an event to happen. There is a time period given to the processor to be in block state and if this time period is over, the processor return to executing state and execute it, but with error conditions there is a unique process id in form (cpu #, process #), where it can be referenced on a system where this lead to the message system, which provide failure tolerant method for interprocess communication.

The system is designed in such a manner that an error will only occur if the sender and receiver processes fail. If any bus error occurs during the process it is automatically corrected in such a manner that it is transparent communication process and logged on the system console.

If the server processor wants to check for any of the messages it calls LISTEN. LISTEN returns the first message from the queue if else it indicates that there are no messages in queue.

The processes in the server will make use of three main functions called writelink, readlink and breaklink. Readlink function is used on the server side and writelink, breaklink are used on the receiver side. Control blocks are kept in reserve to handle deadlocks.



Message system primitive operations

Fig 3

Process pairs are formed for each and every function that is carried out with respect to the use of the web service. Here a process pair implies a primary process and a backup process for a particular function. The backup process is updated on a regular basis by the method of checkpointing which brings the backup process also in the same state as the primary process and so that the backup process can takeover and complete the particular task if the primary process fails. Like a person is withdrawing money using the web service and somehow the primary process carrying out this transactions goes down then the backup process can takeover and complete the task and no inconvenience is caused to the customer.

Then system processes are also a part of the system architecture. System process is the combination of a process with another process called the operator. These are designed using highly complex algorithms. Its main task is to deal with any problems related with the resources being used by the processes at anytime during the web service. Here each process consists of a memory management unit and a monitor process to carry out processes like allocation of physical memory on disc and deletion respectively.

An application process interface also exists due to presence of the tandem 16 which provides the web service application with a library of procedures to access any system resource.

The system is initialized via a cold load from a disc on system. The load file has an image of the operating system, resident code and remaining processes in initial states. At the time of reload the reload command is used. At this time all systems are notified about which processor is being reloaded and a process can be reloaded after repair back into the same processor by using the same command.

Some software checks are carried out to check the operating system for any errors. One of them is done by sending messages between the various processors. Taking an example like if some processors send messages to one processor and all receive the reply I am alive it gives them a confirmation that it is working and if not it is assumed that it is not working.

FAULT TOLERANT STRATEGIES:

There are two fault tolerant strategies used:

1) Check pointing and
2) Application structuring

1. Check pointing:

   The way in which it works is that it makes use of the combination of a primary process and a backup process, which is process pair primitive.

   In this the primary process keeps updating the backup process each time when it gets modified on a regular basis, so that both the processes remains in the same state and if by some unforeseen circumstance the active process goes down then the back up (spare) process can easily takeover from it and complete the assigned task successfully

2. Application Structuring:

   In this strategy the certain part of the application is assigned to a server process pair. At some time multiple copies can be created of server program and can run simultaneously in parallel with other.

   This strategy is of very much used in case of the RPL application, as in this application various requests will be coming in simultaneously like access to account, checking the balance of the customer, and sending the songs to different customer computers. Each customer will have given defined number of process pairs and on verifying the initial requests the process pairs will direct it to the possible server process pair.

<u>STRENGTHS:</u>

Strengths of the RPL server using the tandem 16 computer are:

- It makes the RPL server more reliable.
- It makes the application a real time fault-tolerant.
- It can also have multi processor, so that it can be faster.

<u>WEAKNESS:</u>

Weaknesses of the RPL server using the tandem 16 computer are:

- If the process is waiting for an event to occur and that event is not happening, watch dog timer checks the time for the process and if that process time exceed then that process is executed with error conditions, which is a drawback.

CONCLUSION

In the synthesis of preexisting ideas lie the innovative aspects of guardian and not in the new concept introduction process and message are the notes on low level abstraction all processor boundaries can be hidden from application programs and operating system for the system's to tolerate failure the key is initial abstraction.

In order the system and application to run over a wide range of system sizes that are available, it is necessary to provide it with configuration independence. The extremely general approach to process structuring, interprocess communication and failure tolerance these application communicating process and programs are provided by guardian, there is much talked about structuring programs which uses multiple communicating processes, but there are only few operating system that are able to support such structures

Finally to a large degree the design goal has been met systems that are been installed with processors between two to ten are running online applications they are recovering from failure as well as failures are been repaired online.

References:

Assignment material provided

THIS PAGE IS NOT USED