An Object-Oriented Distributed Debugging Event Model

CHEN Wen-Yu，SANG Nan，QU Hong

(School of Computer Science and Engineering, UEST of China Chengdu 610054)

Abstract For the course of even-based distributed debugging, a great deal of event messages must be collected and handled. This process is not only complicated but also can disturbs the real execution of distributed program. This paper introduces an Object-Oriented event model ,which simplifies the collecting process of event message, makes it convenient to organize and manage the event message database, and adjuste indeterminacy，and reduces the disturbance of debugging to program execution.

Key words distributed debugging; object-oriented; event; replay; indeterminacy

分布式对象调试中的事件模型

陈文宇，桑 楠，屈 鸿

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】针对事件的分布式程序调试过程中，需处理大量的事件消息，如果处理不当，则会影响分布式程序的执行，提出了一种分布式对象中的事件模型，采用这种模型，可以大大简化事件消息的收集和处理，解决了分布式实时计算中的不确定性等问题，并降低了分布式调试过程本身给程序执行的影响。

关 键 词 分布式调试; 面向对象; 事件; 再现; 不确定性

中图分类号 TP311 文献标识码 A

With the rapid development of computer system, hardware is getting cheaper and cheaper. It is getting more and more practical to organize several computers into a distributed system so as to solve difficult computation problems. However, every execution of distributed program can hardly be replayed due to concurrent execution of distributed program; besides to pause or stop any process will have great influence on the execution of others. Therefore, the traditional sequent debugging methods cannot meet the requirement of debugging distributed program, and a special distributed debugging method and tool is urgent need[1].

Event-based debugging can be divided into many types, such as BA method[2], which is based on the snapshot of program status gained during the execution period; BUGNET method and Instant Replay approach[3,4], which is based on the reproduction of program execution, etc. However, no matter which method is applied, in order to determine the execution process of distributed program, we must collect and handle large number of event message .The time which this process takes influences the execution status and outcomes of distributed program.

Received date: 2003 – 10 – 20

Biography:Chen Wenyu was born in 1968. He is a doctor at his post and now he is a vice-professor. His current research interests include the lattice calculate and formal language, etc.

收稿日期：2003 – 10 – 20

作者简介：陈文宇(1968 –)，男，在职博士，副教授，主要从事网络计算和形式语言等方面的研究.

Therefore, it id a very important topic in distributed debugger research to provide a good event message processing strategy in order to reduce the disturbance of debugging to program execution.

An object-oriented (OO) event describing strategy is put forward in this paper. The application of OO strategy to collecting and handling event message can simplify message processing. This method has been used in distributed debugging tool CASE-Distributed Data Base(CASE-DDB) which is developed for distributed C++ language[5].

1 A Brief Introduction to CASE-DDB

Distributed C++ language is developed by extending distributed processing mechanism of OO program language C++. Process class type, process group type, process creating and canceling mechanism have added to distributed C++. It also has message communication mechanism such as entry call, accept, and select by using the language. It is more convenient for us to describe problems in distributed application.

In order to debug distributed application program written by distributed C++, event-based distributed debugger CASE-DDB is developed, which divided into two types all events occurring in the execution of distributed program: global system state (GSS) and inter process state (IPS). The former, is related to status of distributed system, cannot be replayed when distributed program rerun; the later, limited within only one process, is similar to traditional sequent debugging methods.

The collecting and handling of event messages plays an important role in CASE-DDB. All debugging is based on these messages. The quality of event model has a great influence upon the collection and use of event messages.

2 Collecting of Information

The collecting of information references the model of message passing of the process communication. we can get the message of process communication with monitoring the communication. First, we define some source events, such as creating process, page failure, sending message and accepting message, etc. The event information includes the process states with the given time, system queue states, scheduling sequence and process communicating information. These information can be divided into four event types: accepting message, scheduling process, storage swapping and change pages.

We define a standard inter-process communication (IPC) source set，includes Send, Recv, Reply, as shown in figure 1.

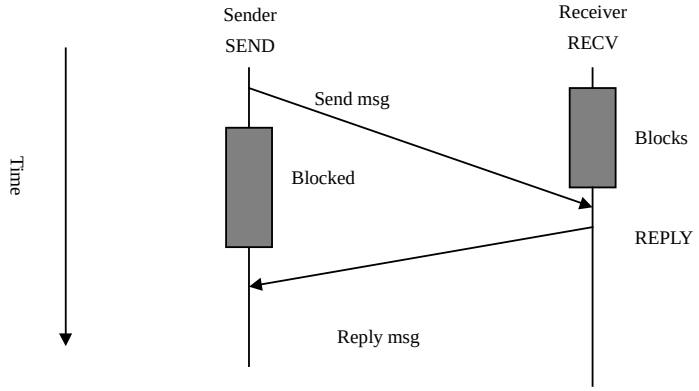

Fig. 1 Process communication

The process communication is based on the passing message，passing message may be synchronism, or may be based on queues. The system states include IPS(Inner-Process State) and GSS(Global System State). Every

process IPS includes the values of local variables and local program counter. Every process GSS includes the process state transform.

3 Definition of Events

As far as distributed debugging is concerned, an event is a kind of system activity describer that can be monitored and can provide multi-level abstraction. Primitive events are the lowest level, which are the lowest level system activity. According to recursive definition, other events may be defined by primitive events and those have been defined.

The changes of distributed program execution status are completely reflected by events occurring during the process. Therefore, all the user's requirement to debugging are to monitor all kinds of events.

4 Object-Oriented Event Model

There are two important facts that influence event-based debugging. The first is event description, which determines event characteristics so as to describe accurately the changes of system status. The second is collecting and handling of event messages, which identified accurately the event when and where to occur, indicates the event messages that are related to the user's view, so as to process respectively in order to reduce disturbance to debugging.

All kinds of events share a lot in common and have some potent relationship among them. If we handle individually, it is difficult to describe these characteristics and to extend event description. As a result, a kind of OO event is put forward.

Every kind of events is defined as an object class, which only describes unique characteristics of this event and correspondent operation. Other features and operation, based on object inheritance, derive from other classes. So event class structure can be defined as

```
event class :parent class-1,…, parent class-n
  {
    private event attributes and operation
    protect event attributes and operation
    public event attributes and operation
  };
```

The model has many advantages. 1) When adding a new event primitive, we only care how to handle its unique build the relationship among these attributes and operation. 2) With the mode, it is easy to create and maintain the message database of event itself and to database. Besides, it is convenient to collect event message. 3) According to special requirements to debugging, it is necessary to single out what we care from a great deal f event messages. If OO model is applied, this part of work can be done completely by handling those relevant classes. 4) In debugging, dome state of variables and functions is what the user cares.

Furthermore, this model improves debugging tools portability and extensibility. For different distributed languages and environment, what we need to do is to add correspondent event primitive to event chain so as to monitor and debug, the whole process is the same.

5 Realization in CASE-DDB

With OO event model mentioned above, CASE-DDB describes all types of event primitive so that all kinds of operation of event messages, such as collecting, arranging, monitoring and replaying, are included in realized event classes. When event classes are realized, private attributes and operations describe their unique characteristic,

protected attributes and operations are what several event classes share, while public attributes and operations provide events with interface to distributed environment. For example, the derivative chain and relevant message of Entry event corresponding to Entry call statement in distributed C++ are shown in table 1.

Table 1 Entry call event class attributes

Derivation Chain	Event Message
Event	Time, Node File
GSS Event	GSS Event Control Message
Process Event	Process Class Message
Process Creation	Process Creation Event Message
Entry Call Event	Entry Call Private Message

As far as the user is concerned, event messages are collected and handled by public member function in Entry event call.

6 Conclusion

In this paper, we put forward an OO event model, the prototype of which has been realized with C++. It is used to describe and handle events in CASE-DDB. This method makes it convenient to debug, transplant and modify debugger. So it is helpful to design debugging tools of other distributed languages or common debugging tools.

Event-based distributed debugging tools are still under study, and researched abroad focus on models. So we must make more efforts to continue its research.

References

[1] Marinescu D C, Lump J E, Casavant T L. Model for monitoring and debugging tools for parallel and distributed software[J]. Journal of Parallel and Distributed Computing, 2002, 36(6): 171-188

[2] Mcdodell C E, Helmbold D P. Debugging concurrent programs[J]. ACM Computing Surveys, 2001, 21(4): 198-205

[3] Leblance T J, Mellor-Crummey J M. Debugging parallel programs with instant replay[J]. IEEE Trans.on Computer, 2001, 36(4): 471-490

[4] Bates P C, Wileden J C. High_level debugging of distributed systems: The behavioral abstraction approach[J]. Journal of System and Software, 2002, 22(4): 255-264

[5] Sang Nan, Gong Tianfu, Chen Wenyu. CASE-DDB: An event-based distributed debugger[R].Chengdu: University Electronic Science and Technology of China, 2002

编 辑 熊思亮