CNIC-00599
ASIPP-0031

中国核科技报告

CHINA NUCLEAR SCIENCE & TECHNOLOGY REPORT

一个数据采集系统用的灵活模拟软件

A FLEXIBLE MODELLING SOFTWARE FOR DATA ACQUISITION

原子能出版社

中国核情报中心
China Nuclear Information Centre

CNIC-00599
ASIPP-0031

一个数据采集系统用的灵活模拟软件

舒炎泰　　陈允惠
杨松琪　　刘根成
（中国科学院等离子体物理所，合肥）

摘　要

这个数据采集系统用的灵活模拟软件是一个基于事件驱动的仿真器。它可用于仿真各种以开路队列网络为模型的系统。此软件的主要特点是它在评价各种脉冲方式或非脉冲方式工作数据采集系统时的灵活性。其灵活性特点还表现在用户可以选择模型中处理器的数目以及每个工作通过模型时的路径。处理器的服务速率可自适应调整，仿真器有流水线控制机构，工作被分割成为若干段来处理，处理器可以代表一个数据压缩单元等等。该文也描述了一些模拟技术和此软件在国内外一些等离子体物理研究所中的应用。

A FLEXIBLE MODELLING SOFTWARE FOR DATA ACQUISITION

Shu Yantai Chen Yanhui
Yang Songqi Liu Genchen
(INSTITUTE OF PLASMA PHYSICS,
ACADEMIA SINICA, HEIFEI)

ABSTRACT

A flexible modelling software for data acquisition is based on an event-driven simulator. It can be used to simulate a wide variety of systems which can be modelled as open queueing networks. The main feature of the software is its flexibility to evaluate the performance of various data acquisition system, whether pulsed or not. The flexible features of this software are as follows: The user can choose the number of processors in the model and the route of which every job takes to move the model. The service rate of a processor is automatically adapted. The simulator has a pipe-line mechanism. A job can be divided into several segments and a processor may be used as a compression component etc. Some modelling techniques and applications of this software in plasma physics laboratories are also presented.

INTRODUCTION

In plasma physics experiments, the data acquisition and processing systems are becoming more important as the data gathered and the complexity of the experiments continue to grow. Though a variety of queueing network tools have been developed, problems to evaluate the performance of pulsed data acquisition systems remain. Plasma physics experiments typically operate in a pulsed regime: the performance of the data acquisition system cannot be evaluated by steady state analysis. Thus it is, valuable to develop methodologies and tools for the prediction of the performance of data acquisition systems: first, to allow system designers to select among alternative systems, and second to gain an understanding of the behavior of an existing system in order to use it best.

Models can be divided into simulation models and analytical models. Indeed, with few exceptions, complex systems are analytically intractable and numerically prohibitive to evaluate. Thus, there is a demand for simulation models to investigate those are as which are not accessible to analytical models. We have developed a modelling software-RTSS(Real Time System Simulator) simulator at the Institute of Plasma Physics Academia Sinica (ASIPP) for simulating data flows in pulsed data acquisition systems. This work is an extension of the modelling of the JET Joint Undertaking CODAS data acquisition system. This simulator also has been used to simulate the data acquisition system of the HT-6M tokamak of ASIPP. Another paper will describe how such a tool can be used for the architectural choices of newer systems at JET and ASIPP.

Simulation systems classically fall into two distinct categories. They either work on a fixed time unit basis or on an event-driven basis. In the *time-driven simulation* simulated time advances in fixed increments. In the second method *event-driven simulation* rather than scanning hundreds or perhaps thousands of fixed time units to discover the next significant change, the event-driven simulator will automatically skip to the next known event, thereby saving a significant amount of computer execution time. Event-driven simulation has a large potential speedup over time-driven simulation. Our modelling software is an event-driven simulation system. Its main feature is its flexibility to evaluate the performance of various data acquisition systems, whether pulsed or not.

In the following paragraphs we give an overview of the simulation technique, our accomplishments to date, and our future plans. We hope this approach will play a role in the design of new data acquisition systems.

1 MODEL

A model of a data acquisition system, and in general of any computer system or

computer network system, usually consists of two parts: the description of the system, and the definition of the workload under which performance prediction should be obtained.

The important element in the model development is the choice of the level of abstraction in the analysis. Normally, we have three possible levels of abstraction:

(1) Hardware level —— Basic building blocks compose the system model, for example, the representations of integrated circuits.

(2) Functional level —— Basic building blocks are the representations of basic hardware units, such as processors, memory and buses.

(3) System level —— Basic building blocks are the representations of complex hardware / software units, such as I / O units, communication units and programs. Both the functional level and system level representations seem appropriate.

The basic model of a data acquisition system is an open queueing network (open in the sense of permitting external arrivals and departures) shown in Fig 1. The open queueing network is a network of interconnected queueing facilities i. e. queueing systems. Every queueing facility consists of a single queue served by a single server. From the outside world, various sequences of customers enter the network. In the network, customers move from one queueing facility to another until they depart from the system at various points For the sake of clarity in our model, we use the term *processor* instead of server in queueing theory, and *job* instead of customer. In our model, the term queue means only the *waiting room* of the queueing facility.

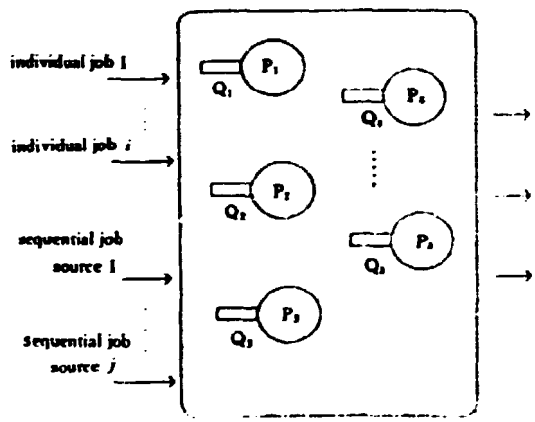


Fig. 1 Simulation model

A data acquisition system model contains several parts. First, we define a set of processors which compose the data acquisition system model. The processors can represent hardware units in a data acquisition system, such as CPU, memory, disk, CAMAC and other peripherals, or software units in the system, such as an acquisition program, a cyclic redundancy check program, data compression and transmission programs. On the other hand, a processor can represent a pseudo unit other than real hardware / software in a system, and it is only a gate to control the job flow in the

model. The modelling software stores the processor parameters, such as service rate, parallel service capacity etc. - in a set of arrays *the processor table*. Each processor has his own queue to store waiting jobs. The queue discipline in our model is: highest-priority-first-serve and first-come-first-serve on same priority.

The second, but no less important, part of the data acquisition system model is the characterization of the workload (i. e. jobs) which applies to the system. In the model, we may assume that a job represents a small or a large amount of data. The job parameters, such as job arriving time, job length, data compression factor of a job etc. - are stored in a set of arrays named *the job table*. The job parameters can be either deterministic or probabilistic. A deterministic model is suitable to the case where the data collection process is completely known. But, the use of simple probabilistic models may provide sufficient accuracy in the performance evaluation of systems during the early design stages.

Finally, the relationship between the processors (model components) and the jobs (workload) must be specified. To define the relationships among the service time, job parameters and processor parameters in the model the user defines several subroutines with standardised interfaces. The relationship among the above parameters may be made as complex as necessary to represent closely the real system by writing specific functions. Another set of arrays, *the job route table*, is used to describe the routes that every job takes to move from one processor to the next until it departures from the system.

2 FLEXIBLE FEATURES

The RTSS simulator has many flexible features, and thus can easily simulate a variety of data-acquisition systems at different abstraction levels. As described in more detail below, we use only 3 different types of events and thus three unified event service routines can be used to serve all events when jobs move through various processors. We can also choose the number of processors in the model, and the route which every job takes to move through the model. This paragraph describes the other main features of the simulator.

2. 1 Service rate adaptation

Processors in the RTSS model can serve several jobs in parallel. The service rate of a processor is a function of the number of jobs which are moving through the processor. The service rate of a processor is automatically adapted down or up, according as the number of jobs in the processor increases or decreases when simulation progresses. This service rate adaptation work is done by two event service routines (*incoming routine* and

outgoing routine).

2.2 Pipe-line control

RTSS simulator inciudes a pipe-line control mechanism, thus can simulate a variety of systems with or without pipe-lining. The pipe-line control mechanism consists of one element in the processor table (i.e. parallel service capacity of a processor) and two elements in the job route table (i.e. two release processor numbers). When a job enters a processor, the processor is locked i.e. its parallel service capacity is decreased. When the parallel service capacity is equal to zero, the processor is busy and can not serve until a job exits. When a job exits from a processor, two processors corresponding to the released processor number are released i.e. their parallel service capacity are increased. Users may define these three pipe-line control parameters to describe various complex pipe-line structures, from pure serial to pure parallel.

2.3 Job segment

Instead of a complete job, a segment of job (i.e. a part of job) becomes the basic job processing unit in the RTSS simulator because typically data acquisition systems operate on limited size memory buffers. A job is divided into several segments and at any given time only one segment can move through a processor. The job segment can be a constant or a function of the processor number and / or of the job number. On the other hand, if a job in a processor queue is less than one segment, it must wait for the accumulation of following parts of the job (this happens for instance when a compression phase is introduced, where the output is smaller than a segment). This job segment feature allows the RTSS simulator to be used when the data amount varies as the jobs proceeds through the processors.

2.4 User interface

The RTSS simulator uses a standard text (ASCII) file interface and a standard subroutine interface. Users use any text editor to build the input files of the simulator which reads all model parameters from these file. It produces several text output files, which can be used for the post processing of simulation results by the user's own analysis, statistical and graphics programs. On the other hand, the subroutine interface of RTSS simulator allows the sophisticated user has more flexibility in defining a variety of random components of the model, complex relationships among jobs and processors, and various stopping rules of simulation.

2.5 Data compression components

Data compression techniques have been quite popular in data acquisition systems because the data volume grows very rapidly as the physics research goes deeper! In the RTSS model, after a job moves through a processor representing a data compression

component, the job length is automatically reduced according to the compression factor.

3 SIMULATION DESCRIPTION

The RTSS smulator, as illustrated in Fig. 2, consists of an initialization routine, a control routine, three event service routines and a report routine. The initialization routine reads all parameters of the model and the workload from user text files, and fills the processor table, job table and job route table. Then, the initialization routine triggers the simulation by scheduling the arrival of all individual jobs and the first job of every job sequence.

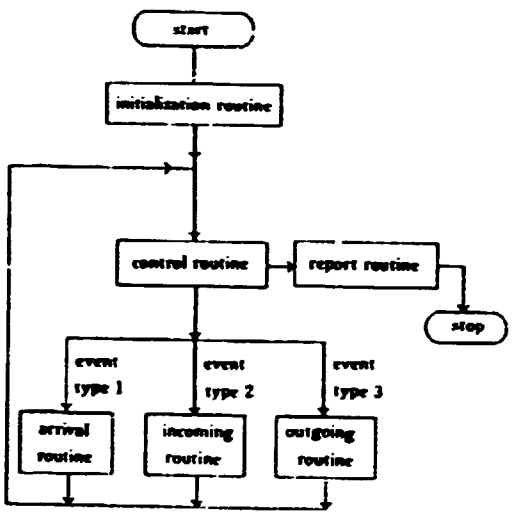


Fig. 2 RTSS simulator

The RTSS simulator is driven by the occurrence of a series of event. An event represents a change in the system state. All events can be divided into three types as following:

Type 1 marks the arrival of a job in the system

Type 2 corresponds to a job's entering into a processor.

Type 3 corresponds to a job's exiting from a processor.

The event service routines are the arrival routine, the incoming routine and the outgoing routine respectively. There is an event list to drive the simulation process. Each event has four elements in the event list. One element is an event identifier which is used for indentifying the event type. The second element is the event time. the third element is a pointer to the job table. The forth element is a pointer to the job route table. From the job table pointer and the job route table pointer, an event service routine can find all the other required parameters from the job table, job route table and processor table.

The control routine is the heart of the simulator. Its main loop is executed once for each event handled. At the start of each loop, the control routine looks up the earliest event in the event list to select the next event. Then, the control routine advances the simulation clock to the event time, removes the event from the event list and calls the corresponding event service routine. The event service routine performs the required

processing for the job, determines its next event, calculates the event time and inserts the next event in the event list. Finally, the event service routine returns control to the control routine after it exits. The three event service routines are described below in detail.

The arrival routine corresponds to event type 1, and does the following things:

(1) The next event (type 2) for the arriving job is inserted in the event list.

(2) If the arriving job belongs to a job sequence and it is not the last job of the sequence, a new job of the sequence is generated and inserted in the job table.

(3) The event (type 1) for the new job is inserted in the event list.

The incoming routine corresponds to event type 2, and simulates a job entering a processor; it does the following things:

(1) If the processor is busy, the input job enters a processor queue.

(2) If the processor is not busy, it serves the input job. The next event (type 3) to occur for the input job is inserted in the event list.

(3) The states of the job and processor are updated.

(4) Because the processor serves one more job, its service rate should be adapted down and the next events to occur for all jobs in the processor should be adjusted.

The outgoing routine corresponds to event type 3, and simulates a job exiting from a processor; it does the following things:

(1) Because the job has finished being output on the processor and the processor serves one less job, its service rate should be adapted up and the next events to occur for other jobs in the processor should be adjusted.

(2) If the output job does not complete, it starts being input on its next processor and the next event (type 2) to occur for the output job is inserted in the event list.

(3) Two processors are released by the output job, according to the release mechanism of the model. Then, the queues of the two released processors are examined; if there are jobs waiting, the one which has the highest priority is selected from the queue and scheduled to be served.

A group of variables describes the simulator state when the simulation clock is advanced by the occurrence of a series of events. The report routine records these variables, so that analysis and statistic programs give the final simulation result.

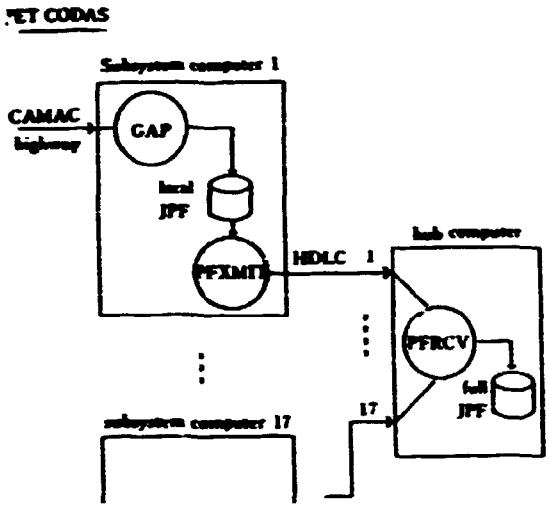
4 APPLICATIONS

In this section, we illustrate the use of the RTSS simulator by three application examples. As the first application of RTSS simulator, a global model of a multi-computer data acquisition system has been established for the JET CODAS[4]. The model has

been used to simulate the data flow with and without data compression compression components in the network system and helped to estimate the benefits of the various possible data compression implementations. It was also used to predict the completion time of the data collection and to identify the bottlenecks in JET planned extensions. The model as shown in Fig. 3, is at a system level i.e. the highest level of abstraction. The 17 computer subsystems of CODAS during data acquisition and collection are modelled with 35 processors. 17 processors represent the data collection programs (GAP), the other 17 processors represent the data compression programs. The last processor represents the data transmission, has parallel service capacity and uses service rate adaptation capacity.

The second application example of the RTSS simulator was to model the data collection on a single subsystem of JET CODAS completely to evaluate the best performance achievable by a pipe-line software architecture. The model shown in Fig. 4, describes the data collection program (GAP) split into two processes. In this case, one process includes CAMAC driver call overhead and CAMAC I/O, while another process includes data packetizing and disk I/O. The model is at a functional level (intermediate level) of abstraction. It consists of 5 processors. Three of them respectively represent CPU, CAMAC I/O and disc I/O. Other two pseudo processors which have infinite service rates, are only used to describe the pipe-line relationship of two processes.

The third application example of the RTSS simulator is to model the HT-6M data acquisition and processing system (DAPS) in ASIPP[5]. The model has been used to select among alternative configurations of processes in the improvement of DAPS. In particular, the model allowed to predict how many simultaneous real time plots were possible. The simulation results fit the measurements of the DAPS system. The model shown in Fig. 5, describes three processes following the data collection process. They are data writing to disc (job flow 1), real-time data processing (job flow 2) and data plotting (job flow 3). The model is at a functional level of abstraction. It consists of 7 processors. Four of them respectively represent CPU, Disc I/O and terminal I/O. The other three pseudo processors are only used to help the release control mechanism to control data flows.



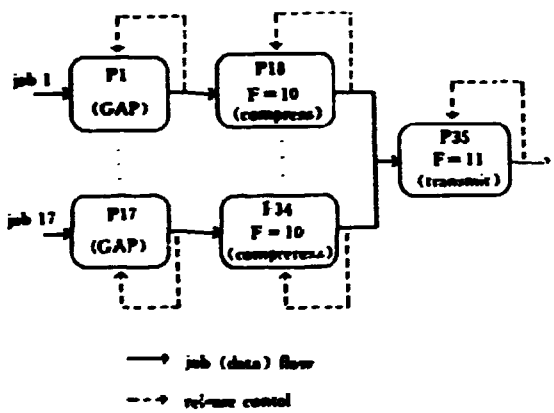


Fig. 3 CODAS system and simulation model

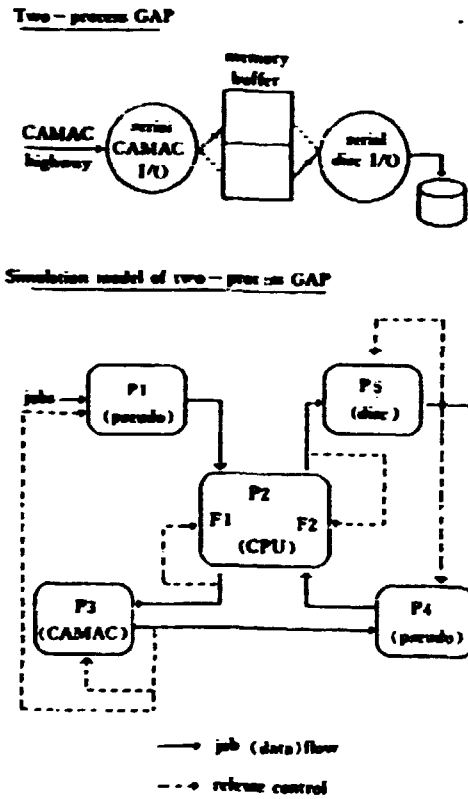


Fig. 4 Example of pipelined data acquisition

5 CONCLUSION

This paper has described an event-driven simulation model of data acquisition systems. The model is written in Fortran and runs on the PDP-11 and IBM PC. For an existing system, the model can be validated by comparison of simulation results with actual measured system performance. This calibration of the RTSS model has been sucessfully done at JET and ASIPP.

Future developments of the modelling software may be expected in a lot of areas, such as having several solution techniques (e. g. simulation and product form solutions) used together in a hybrid solution, capability to define, reuse and share submodels, capability for pre-processing model definitions, capability for doing statistical analysis us-

ing real world data, and more flexible features-interrupt processing, model comments, window interface etc.

Performance evaluation is critical for the design, planning, improvement, and operation of complex systems. Simulation is the only viable means for obtaining accurate performance evaluation for very complicated systems. The application of the RTSS simulator as a tool to choose suitable configurations in JET and ASIPP has illustrated that the proposed model has potential to help solve various design issues of data acquisition systems, such as execution time estimation, data handling, time-out and task allocation etc.

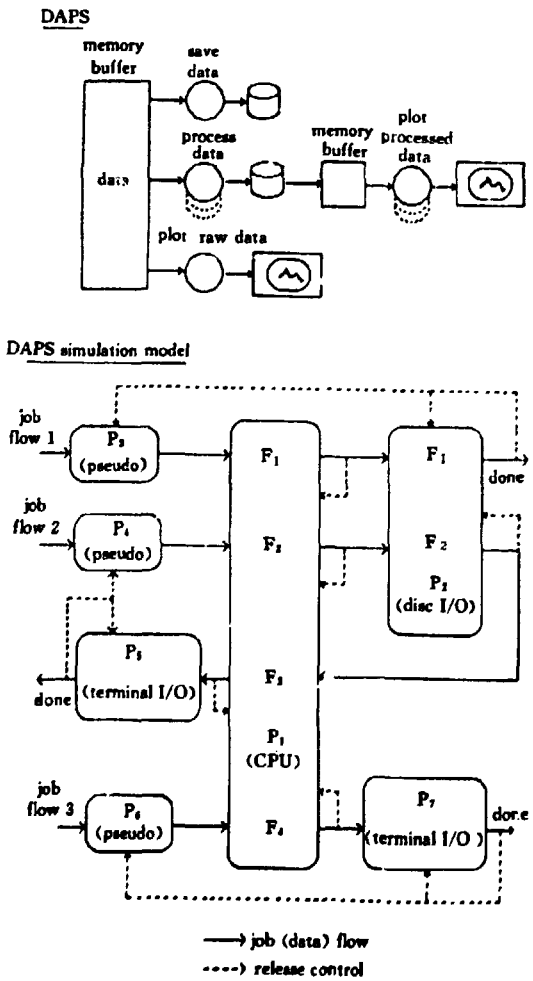


Fig. 5 DAPS and its simulation model

ACKNOWLEDGEMENTS

We thank Dr. H. Van der Beken of the JET Undertaking for his advice and encouragement. We are grateful to Dr. R. Herzog and other scientists of the JET Undertaking for their advice and invaluable assistance in carrying out the performance measurements of the JET CODAS system and the calibration of the model.

REFERENCES

[1] MacDougall M H. "Computer System Simulation: An Introduction," Computing Surveys, Vol. 2, No. 3, September 1970

[2] C. H. Sauer and E. A. Macnair. Queueing Network Software for System Modelling, Software-Pratice and Experience, Vol. 9, 1979, 369~380

[3] F. Neelamkavil. Computer Simulation and Modelling, John Wilcy and Sons Ltd, 1987

[4] H. Van der Beken, H. Clarke, R. Herzog, E. Jones, J. Saffert, Y. Shu, C. Steed, and M. Whearley. Data Acquisition at JET —— Experience and Progress, IEEE Trans. On Nuclear Science, Vol. 36, No. 5, October 1989, 1639~1646

[5] Y. Shu, G. C. Liu, J. Q. Pang, Y. H. Chen, H. Y. Jiang, Data Acquisition and Processing System for the HT-6M Tokamak Fusion Experiment, IEEE Trans. On Nuclear Science, Vol. NS-34, No. 4, August 1987, 782~785

[6] Y. Shu, R. F. Herzog, G. C. Liu. Simulation of Data Compression. JET JDN/H (88) 62, 17 Oct., 1988

一个数据采集系统用的灵活模拟软件
原子能出版社出版
（北京 2108 信箱）
中国核情报中心排版
北京市海淀区三环快速印刷厂印刷
☆
开本 787×1092 1/16 · 印张 1 · 字数 12 千字
1992 年 3 月北京第一版 · 1992 年 3 月北京第一次印刷
ISBN 7-5022-0656-6
TL · 395

CHINA NUCLEAR SCIENCE & TECHNOLOGY REPORT

ISBN 7-5022-0656-6
TL · 395

P.O.Box 2103
Beijing, China

China Nuclear Information Centre