UDC 681.32

L.K. Babenko, A.G. Chefranov, P.A. Fedorov, A.Yu. Korobko, O.B. Makarevich
Taganrog State University of Radio Engineering

Operating System and Programming Technology for Neurocluster Based on NM 640 3 Microprocessors

Neurocluster based on NM6403 neuroprocessors architecture, system software and programming technology are discussed. Special attention was paid to operating system structure; data and control flow between subsystems; internal data structures; system topology; programming language and general parallel programming ideas. This research is supported by Russian Foundation for Basic Research (project № 00-07-90300).

Introduction

Neurocluster based on NM6403 is a part of a whole in general heterogeneous network cluster. System is developed for SPMD (Single Program Multiple Data)-tasks, but it is possible to execute MIMD (Multiple Instruction Multiple Data)-tasks. Such system has to support data distribution, branch synchronization and fast communications between subtasks [1].

NM6403 microprocessor was developed by RT Module [2] and current programming technology is uncomfortable for parallel programming. The main reason of this is using low-level language [3]. There is a high level programming language C++, but it does not allow using vector instructions. Programmer has to compose code for parallel processing, communication organization and other supporting code.

Parallel computing systems need high-speed mechanisms for transferring data and system messages between processes. That is why we use bi-directional communicational links of NM6403. Every processor has two such links with transfer rate of 20Mb/sec. Processors are connected by ring topology using two rings with opposite transferring directions. As an alternative topology we use star topology with communication device in the center. Communication device is based on TMS320C40 signal processors and has memory accessed from neuroprocessors connected to the device. Second communication mechanism is a transferring through PCI or Compact PCI buses, but it is not the fastest way because it depends on number of processors.

Software for this operating system is composed with the help of Fortran or modified version of C. System supports SPMD and MIMD tasks. Applications for neurocluster are image processing, image recognition, maps processing etc.

There are similar systems [4]. Philips produces Lneuro chip, which consists of 16 processor elements with 16 bit registers. Each processor element can work as 16 1bit, 8 2bit, 4 4bit, 2 8bit or 1 16bit processor element.

Hitachi developed Wafer Scale Integration. Every wafer contains neural network with 576 neurons. Neuron has 64 8bit weight.

NM6403 has weight matrix 32x64 bit. Weight bit length varies from 1 to 64. But this processor is not only neurochip, but DSP (Digital Signal Processor) too. It is one chip computer. NM6403 can work without host station. But produced systems and boards based on NM6403 chip have not operating system providing resource management.

We develop providing dynamic resource management operating system for parallel systems based on NM6403 and corresponding programming technology using SPMD mode.

Operating system

Operating system has module architecture and can be logically separated into:

1. Global task manager (GTM), which executed on Intel [5] 80x386 compatible microprocessor. There is only one copy of GTM in system. GTM is a central control subsystem, which monitor and communicate with other system modules.
2. Local task managers (LTM), which executed on each NM6403 microprocessor. It monitors tasks executed on local processor element and communicates with another LTMs and GTM.
3. Remote operating system console. It serves as a communication and control tool between user and GTM.

Dynamic resource management

One of the most difficult problems of parallel operating systems is the dynamic resource management. When system contains more processors than user application needed, operating system should provide ability to relocate system and user tasks from the most used processor element to another (free) one. In this case performance of the parallel system will increase.

When our operating system starts up configuration is detected and stored in system resource table (SRT). Information about executing tasks and their status (running, terminating, waiting etc) is kept in the system process table (SPT). Having these two tables operating system calculates each processor usage and average processor usage. If difference between processor usage and average processor usage is greater than predefined value (system administrator can vary it) then operating system waits for some time interval (reaction time) and reallocation of processes is performed. But in some cases it can decrease total performance because processes are not performed during their reallocation. That is why system administrator has to define reallocation conditions, which depend on the difference between processor usage and average processor usage, reaction time, ability to specify non-reallocated processes etc.

Operating system is able to use hot-swap boards with processors (when CompactPCI bus is used). When user adds board system software adds new resource description to SRT. Adding and removing resources also can be made logically from

console. It means that operating system deletes (or adds) resource description from SRT and stops (or begins) using of corresponding resource.

Programming technology

Programmer can divide code on branches, which will be executed on separated processors. Some branches can be executed in parallel. Others should wait for some event. It depends on logic of task. Following construction shows how branch can be declared.

branch proc_type branch_name (parameter_list) { body },

branch is a keyword, which allow to generate special code for load, unload and communication. Proc_type sets type of microprocessor, which allow to execute branch; it can be nm for NM6403. Branch_Name is a unique identifier of branch. Parameter_list is a list of parameters, which passed to branch; body - is the set of operators and commands.

Example:

```
branch nm Sum (int a, int b){
        gr0 = [a];     // value a
        gr1 = [b];     // value b
        gr0 += gr1;    // sum and result
        pushres gr0;   // save result
        }
```

When OS loads branch, process is created, which consists of set of segments: code segment, data segments, stack segment etc. Parameters passed to branch and results of calculation are pushed to stack. When process is terminated, result is returned to main program.

Also parameters and results can be passed with help of MPI [6] (Message Passing Interface) or MPI-RT [7] (Message Passing Interface - Real Time).

Branch can be loaded by branchstart routine. Declaration of this function is following:

```
handle branchstart ( branch, num_param, param_list);
```

branch - identifier of branch , num_param - number of parameters, param_list - list of parameters. Function branchstart returns the process descriptor (ID).

Branchwait routine is used for barrier synchronization. Parameter of this function is a list of process descriptors terminated by 0 (null).

Function branchkill terminates execution of process. Process descriptor is passed as a parameter.

For running SPMD-task, which consists of num_branches branches with the same code and different data, next function is used:

```
handle SPMD_start (num_branches, branch_name, num_param,
list_param [dist]);
```

branch_name - branch identifier, num_param - number of parameters, list_param - list of parameters, dist - kind of data distribution. Function returns descriptor of SPMD-task.

For barrier synchronization with termination of SPMD-task following routine is used

```
SPMD_wait (handle ID);
```

ID - descriptor of SPMD-task.

Commutation

During the first stage of OS staring detecting of configuration is performed. OS loader sends test message to first processor, which was found. When processor receive message, it "remembers" sender and sends test message with help of own links to another processors. This procedure repeats while there is one (or more) processor, which has not received the message. Then message is returned back and every device adds own identifier and number of link to the end of message. Received sequence is analyzed to build topology of system.

In that case when there is no special commutation device, communication is performed by using host-computer. It means that message is sent to host-computer and then host-computer sends message to target processor. This method of communication is slow and can be used when communication is performed rarely.

If there is special commutation device in the system, then all messages are sent to that device and later to target processor or another commutation device.

When program is compiled compiler checks data requests, which are placed in RAM of another processor. In that case compiler adds code for accessing to that data. Host-computer (or commutation device if it present) has table of data location, which are used during search of data.

Conclusion

Main operating system modules were coded and tested. These modules are Global Task Manager, Local Task Manager and system console. Technology of programming is ready.

Next part of work will include development of communication libraries, system drivers, C and Fortran [8] compilers, system tools and such applications as image processing, image recognition and maps processing (e.g. fingerprint recognition). Also we plan to develop technology for converting linear programs to parallel ones. This system will give results as graphs of execution, where nodes are independent execution units and links are data transfers [9].

Literature

1. Korobko A.Yu., Chefranov A.G. Operation System for Computing System Based on NM6403 Neuroprocessors. In proceedings of conference «New Information technologies», 2000, Taganrog. – P. 130-132 (in Russian).
2. RC «Module» web-site, NM6403 Documentation, www.module.ru.
3. Neuroprocessor NM6403. Base Software. Assembler Language Description. UFKV.30002-01 35 01, RT Module, 1997. – P. 83.
4. Korneev V.V. Parallel Computing Systems. Moscow, Knowledge, 1999, 320 p. (in Russian).
5. Intel Corporation web-site, www.intel.com.
6. MPI-2: Extensions to the Message-Passing Interface, Message Passing Interface Forum, July 18, 1997.
7. Document for the Real-Time Message Passing Interface (MPI/RT-1.0), Real-Time Message Passing Interface (MPI/RT) Forum, March 6, 2000.
8. H. Zima, P. Chapman, H. Moritsch, and P. Mehrotra. Dynamic Data Distributions in Vienna Fortran. In proceedings of Supercomputing `93, Portland, OR, November 1993.
9. S. Mitrovich, S. Murer, A Tool to Display Hierarchical Acyclic Dataflow Graphs. In proceedings of the international conference Parallel Computing Technologies, Novosibirsk, September 1991. – P. 304-315.

Материал поступил в редакцию 15.06.01.