Emulation of Complex Computing Systems and the Software Transferring Process

A. Sourkov
E.P.A. Ltd.
St.-Peterburg, Russia
Shura@comcit.spb.ru

V. Urusov
E.P.A. Ltd.
St.-Peterburg, Russia
kondor@mail.marinform.ru

Abstract

The present article describes an intelligent (programmable) hardware emulation and some questions concerned with transferring data and executable code from such systems. We will pay attention only to those computer complexes which peripherals can fulfil their own special instructions and non simplest actions, without any request to central processing unit **(CPU)** of these computer complexes. Right now IBM/360/370 is a classical example of such complex. Its I/O subsystems have own embedded programming language (named as channel programming language) and its own CPU and address translation independent path to memory. This way eliminates the information streams bottleneck which is known as system bus and increases the computer complex productivity. These peripherals can make hardware search with a pattern (hard disc subsystem), some transaction with users(terminal subsystem) e.t.c. Unfortunately, the modern computer industry tends to increasing throughput of a single unit - CPU at the expense of oversimplification the others. The bright example is 'winmodem' - the unit which cannot put a byte to phone wire without dedicated CPU service subroutine.

1. Introduction

There are many cross programs which are emulating one CPU using another. It is simple. Our specific features (which has not developed at our opinion yet) are peripheral devices emulation which use input-output simultaneous to main calculating process, multiport memory subsystem and a complex emulation of such systems (CPU+ peripherals+ memory). We have our own software product - the IBM/360 emulator which was tested under some mainframe operation systems and applications on some hardware platforms under some modern operation systems.

The emulation technology gave us some nonobvious methods. We have more complex, detailed and customised tracing technology now. We can adjust input-output flow on our own. We can see data within instruction fulfilling (what may be rather convenient and important in the case of chain or block instruction). We can make traps in multiprocessor configurations e.t.c.

In practice we can in a great possibility catch date requests which follows us to 2000year problem solution. IP technology allows us to make a remote mainframe system console. These consoles can be gathered and a single operator can manage some several distant mainframes. This case make cheaper the mainframe maintenance. Also we can place users' terminals anywhere.

The practice aim is data transfer from naitive IBM/360 format to modern personal computer (or RISC) format and a support of some transfer time critical industry and military processes still using mainframes and cannot be shut down to update its software.

There are four ways of software updating to a new platform: reengeneering, rewriting, recompiling sources and emulation old environment on a modern one.

Our technology (environment emulation) helps enterprises which haven't sources of their working programs, its authors and have dim algorithmic understanding of these process. This way also have a set of inportant features:

- the task is transferred as 'black box', such approach eliminates security problems and necessity of algorithms investigations,
- the transferring executing process will have no changes which makes old licenses and sertifications still valid (in fact it is equivalent mainframe CPU changing),
- though emuladed system works slower it is compensated adequate modern CPU grouth,

Proceedings of the Workshop on Computer Science and Information Technologies CSIT'99
Moscow, Russia, 1999

- transfered software can be serviced by the same stuff after a little modern system depended training,
- the data can be shared and manipulated using all modern technologies.

The other item to mention is an environment in whitch transferring software works. The customer's interes lies only in applications, so we can make some additional optimisations. Some interesting discoveries are in the article.

So the emulation process has its own special advantages, which are not so sparkling but rather important. By the way, this method can work where the others cannot.

An unprecedently rapid growth and development of computer industry (now computer communication also) is its main feature now. The well known Moors theorem tells that the density of microelectronics chips (and therefore the complexity of built using it equipment) makes twice higher every 18 months. This growth rate leads to the fact that enterprises of another industry branches cannot stand such rush development in organization and/or financial aspects and fall behind technology. The longer gap the more difficult to communicate with modern computerized society. There is a level of this lagging when connection becames impossible. The data on such complexes became unreachable and algorithms are lost.

For example, there were 6-bit in a byte computers (ICL19xx, in Soviet economic block they was known as ODRA ™). Up to date still exists several floating point formats. IEEE issued the floating point standard but many manufacturers still use its own.

Another special group of information consumers are government structures. This kind of consumers also requires special stuff with security admission, certification procedures, fault tolerance and so on. Meeting these requirements lasts for months and when all these problems are settled these complexes find themselves obsolete. The situation is made more difficult with the process nonstop requirement.

Another feature of the rush development is service time decreasing. As a new computer generation comes hardware and system software support of elder complexes decays for a few months. This situation follows to the fact that every trouble may be fatal. In the case of hardware fault the data may be lost and/or the production may be stopped or disorganized in any range. Eliminating of software support follows to a situation when in some time nobody understand how the computer complex works. So changing of data flow becomes impossible. This leads to the fact that owners of such complexes cannot adjust its working to any world or technology changing, the production cannot develop and be fully managed so the decay of these enterprises becomes a question of time.

The fact is the information which was gathered and stored in such complexes is worth to try to use it. In the case of government and/or military information access to it sometimes becomes a state worth question.

We have developed some conceptual approaches to solutions of such kind problems and made appropriate software. Later we shall discuss about the most often happened problem - IBM/360 heritage and how to use and execute it on a modern computer (Intel 80x86, MIPS, Alpha, PPc and so on).

Here we shall discribe the computer system emulation with programmable peripherals (i.e. devices which can fulfil its own special instructions and make some complex operations without direct CPU intervention.

Now we assume IBM/360 computing complex with channel comands interpretator unit, hardware disk search with a pattern e.t.c. Modern peripherals are more simple. Now we have a great advantage in CPU power and a corresponding simplification in other hardware permanent demanding CPU attention. As an example we note 'winmodem' - the device which cannot send a byte itself - all subactions are to be prepared with CPU.

Emulating old systems with modern hardware we couldn't avoid this this problem. We had to go by general way increasing peripheral devices' intelligency with CPU. But we arranged any tricks to preserve native stucture and behavior of mainframe peripherals.

The main important mainframe feature input-output working simultaneousely with computing really doesn't work on a host computer because its architecture doesn't support it. So emulated application using this kind of input-outpwu works slow. But it works. Anothe problem is a real-time mainframe timer. It is a general problem of avery cross system. We didn't investigate this one because really high accuracy of its emulation was not so nessessary yet.

Some pecularities have memoty subsystem. The simpliest mainframe memory unit is dual port unit. There can be two independent memory request at once. During program emulation on a modern host computer this effect is not happen. In this case all processes goes sequentually and an ordinary memory works. Also we can use different page and segment descriptors for dufferent emulated processor units, direct adressess for input-output flow and key memory security.

In our approach we are not limited with hardware size of memory page. Therefore we have no problems of different page sizes of different CPU models.

The emulation technology gave us a set of develops and extensions. We can debug in a greater details and have more conveniences which we can adjust to ouselves. For example, we can debug chain and block data instructions with overlapping areas, we can manage input-output flow and see the data changing within instruction executing. Also we can make address and data traps, trace multiprocessor request and so on. When we search, catch date request and trace following instruction we have a chance to solve 2000year problem in

the traced application. Now we are creating more specialised tool for this task.

Embedded IP technology gave us oppotunity of creating remote mainframe console. These consoles can be gathered on a single personal computer and one operator can manage several distant mainframes. This way makes mainframe maintenance cheaper. Also we can place user terminal anywhere. In this way we use standart IP utility telnet 3270 whitch offers routable worldwide packet interface. This utility has every IP supporinng operation system so user terminal may be arranged under a lot of operation systems including MAC or Amiga for instance. Moreover, they can work as mainframe user terminal for several emulated mainframes.

Real worth aim of this activity is data transfer form IBM/360 format and storage into modern format and carrieers. Also importance has some time critical military and/or industry application support. In these cases required some licenses and sertificates for software. Really now there are not software engeneers who can stand all these processes.

2. The premises of the task

Transfer of exising programms into open systems is a question of vital importance for many organisations developing and using information systems of any kind.

The Gartner Group and International Data Corporation have questioned some computing centres using IBM mainframes. This questioning told that more than 70% were going to transfer their programms and data into open (modern) systems within several next years.

There are four possible ways: reengeneering, rewriting, recompilling sources and emulation old environment on a modern one.

Reengeneering includes forming a new task, methods, which follows to finding new program models and algorithms fully using modern techniks advantaches.

Rewritting includes adoptation old sorces from old data structures to new using known algorithms.

Recompiling includes use old sources with new program tools to perform executable code and acceptable data structures.

And the emulation old environment on a new platform.

Let's see first three ways in details.

Reengineering seems as the only possible solution for some programs. Of corse it is expensive, long and complex work. It demands task produsers, high skilled application specialists, algorithmic creators, programmers, system programmers an so on. In some cases all or a part of the staff are to have special permissions or a something like it. This way is the most reliable and preferable but it is the longest and most consuming way. This way produces a new product. Therefore it in military or similar case has to pass sertification expertise. It still consumes a lot of time.

Rewritting and recompiling are seemed as fast and cheap solution in fields where changes are not so large. But there are some advanteges and disadvantaged.

One can consider as advantage following:

- The result is a native build programm, compiled in native codes and using native operation system requests.
- Quality of workers. This way doesn't require high qualified expensive specialists. All work can be checked under test data what makes the process faster.
- This technology can be applayed to the tasks which have sources but have no algorithmic calculations.

And about disdvantages:

- Architecture features which were used in older solution still force on new product and in some cases will determine its working principles. For example, a program working with index files REGIONAL(X) type in PL/1 language will leak its productivity after recompilling into UNIX or WINDOWS -like operation systems because this type of index files assumes hardware seek with a pattern and was optimised for using this hardware capability. Program recompiling or cheap (low qualified patch) will greatly decrease program effecivency. UNIX version of REGIONAL(x) under plain POSSIX interface is extremely uneffective compared with mainframe analog.
- Low quality engineer can miss some specific stucture updates or something else. IBM/360 has following basic data types:

- 32 and 64 bit integer,
- floating point data with 24, 56 and 112 bit mantissa,
- packed and unpacked decimal data.

Under automatic recompilation or formal rewritting the data format really will be changed leading in incompatibilities between data structures and execution code. All float point will be in general transferred into 64 bit DOUBLE PRESISION REALs. All old floating point arrays become ivalid.

This situation requires rewriting of all working structures before translator runs. After this stage correction will require structure or record pointers and so on. At final most convertors forget the fact that IBM/360 has hidden floating bit in mantissa in FPU so the coreect transferred data produce corrupted result.

The same situation is with fixed point data. As a result we can see that program algorithms cannot work and the situation require manual data transfering. At practice we have manual work under about 30% data and sources and it consumes more than 90% of total time.

We suggest to discuss the fourth way - the emulation technology and related processes.

This tecnology has following pecularities:

- The task transfer does not require any programmer at all, there also may be no profile specialist.
- The task is transferred as 'black box' without any intervension in its working process. In fact it is equal mainframe CPU changing for most tasks.
- The productivity of computing complex is high level decreased but it is corresponded with high level increased modern host CPU productivity.
- The emulated applications can be serviced by the same old stuff. New host system training in this case is simple and may be done for some hours in one of many modern system software training centres.
- Data may be stored on shared disk in known format what makes possible to get access to them using modern computer facilities and appropriate drivers or convertors.

3. IBM/360 to IBM PC emulator as example

We have developed the software for full emulating behavior IBM/360 mainframe on a Intel 80x86 architecture (**the emulator** later). The emulator also makes emulation of mainframe hardware and this thing allows to run almost all IBM/360 designed software.

The 4.12.0 version on Dec. 1997 represents following:

- IBM/360 CPU in full range,
- Disk drives 29Mb, 100Mb, 200Mb,
- Types,
- Printer and reader,
- Display 7920,
- System console.

Realisation:

- Disks and types are emulated as files with internal structure nesessary for its working. We have there tape mark emulation for types, own address and other for disks and so on.
- Displays are emulated on PC monitor
- Printer and reader are represented with DOS files or as task separated files in a DOS folders set. This folders may be network shared folders.

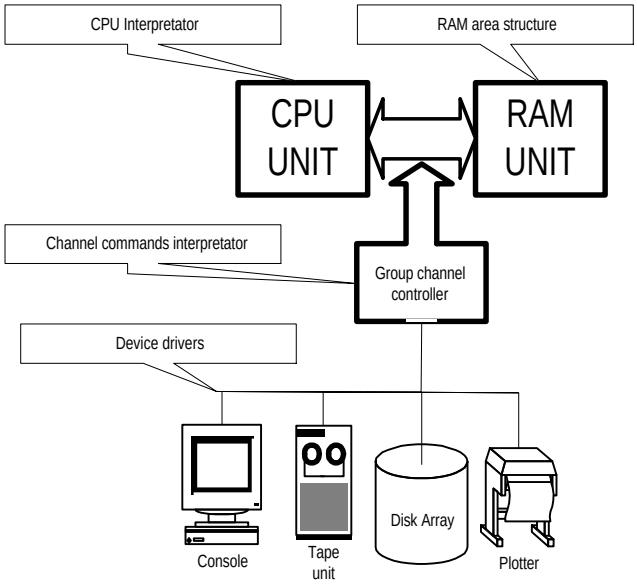

Figure 1. Block-chart of the emulator.

The former raw product later was improved with structure optimisation. It is clear that tasks are run and data are stored not with a kind of magic. Application have a set of program requests derived from local operation system.

We decided that not wise to emulate the whole operation system and applications. Why have we to make emulation of an application request to a operation system, then make emulation of operation system redirection to hardware, hardware responces, preparing this responce with operation system and final answer to the application? It is clear that these requests are mostly simple file or communication requests. Usual requests such as find file, open file, read or write data to file, close file and so on can be translated to similar requests of the host operation system. It has redused a lot of tine and computer resources.

System interface (in this time too) is designed to brimg maximum flexibility. So all transaction were made using system calls, which were traced. These call entries , functions and protocols were investigated and redirectors of some kind requests were placed.

The main problem in this way is the problem of a large varaety of different IBM/360 operation systems. So we didn't include our redirectors in main code.

Now based on a growing practice we desided to try to play with SVM operation environment. In this field we have maximum transferring tasks.

SVM basic idea was to create a set of vitual mainframes on a real one. This set was managed with Monitor of Virtual Machines (**MVM**). User in this virtual mainframe feels himself as in a real one. Virtual mainframe fulfils all user mode executable code as a real one. But separate users had separate (virtual) disks, memory e.t.c. SVM user could make special SVM request to take additional capabilities.

All system calls were serviced with MVM. MVM also translated user virtual input-output channel programs into real channel program, arranged inter- virtual mchines communications.

SVM applications were made under such program environment It was not nesessary to MVM support longer.

We tried to reanimate SVM user program environment under the emulator. On this way we found that these MVM requests were made using separate single supervisor protected really useless IBM/360 CPU instruction. We added only MVM request emulating unit based on single instruction interpretator.

Another large optimisation step was IP protocol realisation. IBM in its operation realisation made IP support over two its own incapsulated protokols. Nobody now use these old IBM protocols. So we cut them without any harm.

Another step in our work was hardware optimisation routines. The most slow part of disk input-output is the hard disk head seeking. So was in mainframe age. Understanding this fact developers of many operation systems added optimisation routines in their systems. These optimisation routines in general sort hard disk task queue to optimise head walking. We decided not to cut this code. But if all host tasks including emulated hard disks were placed on a single disk our decision became useless. So we went furher, we tried to place maiframe disk on a separate low capasity hard disk drive. Well, old technologies still works. We caught a little onput-output flow increasing.

Now the emulator sources are rewritten to get full 32 bit version under C language. This way allowes us to apply this software on a lot of platforms and operation systems. Comparing results obtained on different platforms we'll get new results and new capabilities. Now we are collecting statistical data and improving emulation process. We have no enough data right now but it is to be later.

The data and software transferring process

One of main problem here is a hardware one. There are few opportunities and they don't seem very helpful. For example general communication unit with sequential interface (like RS232) had 300 baud speed (less than 40 bytes per second). It is considered now that more uesful way lies throug producing special controllers. There are a lot of mainframe terminal connection controllers. It can be placed into an ISA slot of IBM PC and it has BNC connector to be placed within terminal line. There are some disadvantages:

- the real throughput is still low,
- nessessity of special protocols and IBM/360 utilities to send binary data through terminal line (alphabetic),
- system digging if we want to have any system information, and permanent adjusting these tools due to differences of many versions and types of operation systems.
- placing special hardware in customer's area what brings security time and some other problems.

The more natural way we thought was using system tools and native IBM/360 carriers. This way minimises our extra work. We decided to use types. Now we don't neet of making old IBM dependent tools. Every system had nesessary tools because type in mainframe is a standart backup tool and every operation system had appropriate backup tools which we can use. Specialised PC hardvare and its driver is enevitable, so we adjust type drive to SCSI interface and this drive settled all problems. This adjustment was easier because electrical ISA parameters are more complex than SCSI.

As a criteria of successful data and software transfer we chose identity of test results obtained on transferred executable code under the emulator with transferred data (real or test) with the same code and data set on mainframe. For greater reliability we fulfil some runs on a different data sets and asked a customer to make extremely hard set (with larger swapping, higher input-output activity, higher floating point presision and so on).

On missing we trace the emulated application. It is cleaqr that troubles can be only in input-output subsystem. We check all mainframe hardware request for intelligence. After detectin suspicious ones we make further investigation, put some questions to customer's hardware engeneers and creating cure for that problem. At the end we made special solution.

4. Afterwords

We still haven't tried emulator under multimainframe operation systems. But it seems possible to realise every CPU module as a separate thred on power RISC computer.

On the way we will solve the common emulated memory problem, make pefix memory mapping, do interprocessors requests e.t.c. The way have just started but it seems interesting. We can emulate separate CPU on separate hardware solution. In this case all synchronising are on emulated multiprocessor operation system expence. But the result is unseen so it is attractive. It is clear that real time tasks are still beyond our activity. But we hope that increasing productivity of modern processors one day makes this kind of tasks possible. Another future way of our technology is transaction computer.

Another our goal is preserve made algorithms. In past in computers lack programs were better. Now one is waiting new faster CPU or larger memory generation.