Disclaimer

This note has not been internally reviewed by the DØ Collaboration. Results or plots contained in this note were only intended for internal documentation by the authors of the note and they are not approved as scientific results by either the authors or the DØ Collaboration. All approved scientific results of the DØ Collaboration have been published as internally reviewed Conference Notes or in peer reviewed journals.

Running a Large Monte Carlo Program in a Farm of MicroVAX–II Computers under VAXELN

David Cutts and Jan S. Hoftun
Brown University, Providence, RI 02912

Abstract

This paper describes the operation of a loosely coupled "farm" of MicroVAX–II computers running in the VAXELN environment. Our application used the GEANT Monte Carlo package to simulate 2 TeV proton–antiproton collisions as seen by the DØ detector at Fermilab. The layout of the farm, the control software, and specific (small) changes to the program which were needed for operation in the farm, will be explained.

Introduction:

The event–based nature of data in high–energy physics lends itself naturally to parallel processing of individual events which may be done in a "farm" of computers coupled together via some loose network to a host computer. This parallel event–processing scheme can be used as part of data acquisition; for example, one may use such a farm for fast online filtering of events, with the nodes capable of running filter programs which are written in high level languages. Such an arrangement is the basis for the DØ data acquisition system which has been described elsewhere [1]. The combined hardware/software solution chosen for DØ is based on Digital Equipment Corporation's MicroVAX processors and VAXELN software package. In order to gain experience with running large FORTRAN programs in this environment, a first try at operating such a farm for Monte Carlo simulations was made in late 1984. The program used then was ISAJET, a Monte Carlo program which generates high–energy physics collsions. This work provided experience in assembling the control programs and other utilities needed for such an operation.

In the summer of 1986, the DØ experiment had an urgent need for a large number of simulated events to finalize the detailed design of the detector and to investigate triggering schemes based on this design. A Monte Carlo program to generate the events

as seen by the DØ detector had been developed based on the GEANT package from CERN [2]. But there was great difficulty finding enough resources to run such a very CPU–intensive program (about 1 hour of CPU–time per event on a VAX–780). None of the institutions within DØ (including three national labs) could dedicate sufficient conventional computing capacity to dedicate to the generation of the required 10,000 events on a short timescale, and because of code incompatibility and other problems neither the Fermilab ACP multiprocessor system nor an outside supercomputer could be used. The collaboration turned to the option of running the program on MicroVAXes under VAXELN. This special version of DØGEANT which was to run under VAXELN turned out to be easily put together; beginning with the VMS version of GEANT we added various control features and with a few weeks of effort had a production version running. Most of the time was spent developing general aspects which are useful for setting up any application to run in the farm.

Hardware Layout:

The basic hardware used in this farm consists of MicroVAX–II CPU's (in either the KA–630 as used in VAXstation–II's or the KA–620 real–time version which may only run VAXELN), external memory (most nodes used in this run had 4MB added to the 1MB on the processor board), and DEQNA ethernet interfaces. In addition to the farm nodes, we used as the host a MicroVAX–II system with 2 RA81 420MB disks and a TU81+ 6250 bpi tape drive. Data transfers between the host and the nodes utilized Ethernet, whose support is automatically included with VAXELN systems. Since data was actually transferred only at the beginning and at the end of each of the very CPU–intensive events, the restriction of the Ethernet bandwidth did not pose any limitations. The farm nodes were housed in various backplane configurations: in custom setups used for data acquisition test work, in rack mounted BA–23 chassis and even in a fully configured VAXstation–II. The basic hardware setup for the GEANT run is shown in Figure 1. For the run described the farm consisted of up to 16 VAXELN nodes.

At present, several innovations are affecting the basic hardware described above. A new custom backplane has been developed commercially [3] for the DØ data acquisition farm and will also be used in a slightly modified version for an offline farm. High speed communication channels are also being developed for use in the host–to–node data transfers. This path (40 MB/second per channel) is based on the dual–ported Q–Bus memory boards [3] devloped for the DØ data acquisition system.

VAXELN:

VAXELN is a "software product for the development of dedicated and/or real–time systems for VAX processors". It is not an operating system, but allows the user to have full control of all services on a target processor. The development of

these application systems takes place under VAX/VMS, using the same compilers, the same linker and all the other facilities for ease of program development found in VMS. The programs may be written in any high level language and linked together to form separate programs. VAXELN comes with its own "dialect" of Pascal, called EPASCAL, which is a superset of standard Pascal with extensions for interfacing to the VAXELN services for multitasking, network services, and real–time resource management and device control. VAXELN also provides such an interface for VAX–C, using the standard VAX–C compiler. FORTRAN is supported with a Run–Time Library which makes essentially all VAX–FORTRAN extensions available under VAXELN, although there are a few restrictions as to some types of I/O operations (use of indexed files), and logical file names are in general not supported.

To run a program in a VAXELN target node, the code is first compiled under VMS, and then linked with the VAXELN libraries. Next, a system file is built with the small, menu–driven facility, EBUILD, included in the VAXELN tool–kit. Such a system file may then be used to download to a target node, and will become the only system running there. Particular consideration has to be given to the system size, as VAXELN does not swap pages in and out of memory (no local disk); the whole image therefore has to fit in memory at once.

The programmer has full control over which services like drivers etc. are included in the system. This makes for a very efficient run–time environment with little system overhead. The VAXELN tool–kit also includes a remote debugging facility, EDEBUG, which runs under VAX/VMS but connects to the remote node and allows the programmer to debug his/her code directly as it is running there. EDEBUG is similar to the VAX/VMS debugger and is fully symbolic, with all variables for example available to the programmer for interactive inspection and possible modification.

Changes Needed to Make a FORTRAN Program Run under VAXELN:

Very few changes have to made in the FORTRAN code to make a FORTRAN program run under VAXELN. Most of the changes have to do with OPEN statements: making sure OPENs have explicit host node and disk specifications. Also, any refrences to system service (SYS$, LIB$ etc.) routines should be removed and/or replaced by calls to VAXELN equivalent routines.

Control Program for Farm Operation, FARMRUN:

The program FARMRUN was devloped to control the running of this farm. Principally, FARMRUN provides server functions for the input and output event streams. These servers run separately as batch jobs, and command files are used to start them and keep track of which files to use etc. An operator controls the farm by running FARMRUN interactively and may then give START, STOP, PAUSE and CONTINUE

commands. The program is also able to interrogate each node and obtain a status message describing the current state of the node. This program uses two other program packages to perform its task. For interfacing with the operator, FARMRUN employs the command/menu package COMPACK developed for general use in the DØ–experiment. COMPACK is a general purpose command/menu interface package which has both a "line"–mode and a full–screen "menu"–mode of operation. The user may switch between the two modes at any time. COMPACK also has a command file mode and may be run in batch mode without modifications. It is written in FORTRAN with a subset in FORTRAN–77 for ease in transport to other machines. The full–screen part of COMPACK uses the SMG$ set of routines to perform screen I/O. To send commands to each of the nodes in the farm and retrieve status messages, FARMRUN uses a small package of FORTRAN–callable routines, ELNCON. ELNCON simplifies the system programming needed to perform DECNET I/O under VAX/VMS and provides a Fortran interface to the VAXELN services for network I/O.

Programs in each Farm Node:

The programs running in each farm node are shown schematically in Figure 2. The main program itself, via the call to the initialization routine, starts a few subprocesses used for control and for message passage. These processes go into wait states and hibernate until signaled by software flags. This technique avoids polling–type statements in the main code. The subprocesses take care of control/status messages to and from the host, and may stop the main program if such a command is passed to the node.

The first program which gets control when a node is booted, is JOB__START, a small control program using the VAXELN feature which lets one program control how and when another program is running. JOB__START is set up to start the main program with certain input parameters and then go to sleep until the main program disappears for some reason (ends naturally or is aborted by operator intervention). When the main program goes away, a new copy is immediately started by JOB__START. This action, which does not involve any downloading of the image (the code was already in memory), cuts down on network traffic when many nodes are stopped to be restarted again with new parameters or for other reasons.

Various utilities were also developed to make the computers in the farm work together and make the operator intervention easy. For control and downloading of parameters etc., a call to a special initialization routine is put into the main program. It returns the DECNET–number of the host node and other program–specific parameters used at startup. The name and the number of the node itself which are useful for setting up file specifications for OPEN statements or setting seeds etc., are also available. In DØGEANT, input and output event files were opened via calls to a special control routine which could access status variables in a common block.

Operation of the Farm:

Figure 3 shows schematically how the farm operation occurs. While several components of FARMRUN (Input Server, Output Server) may be running in batch mode at the same time, only the Input Server is really needed for the farm to operate. The input and output records in DØGEANT operation are simply files on the host containing one input or one output event. A node opens and reads an event when it is ready for another and then deletes the single input event file. The Input Server wakes up every so often to check if the input event file has disappeared in the meantime. If so, it reads the next event off the file of input events and writes a new single event file. When a node is done processing an event, it opens a new file on the host disk and writes the event to it. The Output Server also wakes up every so often to see if any new events have been written in the meantime. If so, it reads in the event, checking that its format is in order and writes it out to a file of collected output events. When a certain size of the output file is reached, the Output Server closes it, and restarts itself using a new file for the next set of output events. Thus collected output events can be copied to tape at any time thereby freeing up valuable disk space. The FARMRUN component which interactively controls the run and provides status messages is started only when such an operation is needed.

Conclusions:

While fulfilling a vital need for the DØ Experiment, we have demonstrated that MicroVAX–II computers running under VAXELN are very well suited for event–based data processing tasks in high–energy physics. The amount of system programming involved to operate such a farm is small and easily managed, even for massive software packages like GEANT. Much of the convenience of our system derives from utilities built for the DØ data acquisition system. Ethernet communications, included in the VAXELN nodes, provides a very natural medium for data transfers. Ethernet bandwidth has proved acceptable for CPU–intensive jobs like GEANT, but larger farms or jobs with less computation per event may rather employ hardware developed for the DØ data acquisition farm. Our experience in using the MicroVAX farm to provide Monte Carlo events for DØ has been highly successful and should be a model for similiar applications elsewhere.

Acknowledgements

We wish to acknowledge Ray Zeller and Chris Johnson of ZRL for their vital contributions to the DØ data acquisition hardware and software and to the operation of the system described here. We also wish to thank Digital Equipment Corporation for their interest and support of our application of MicroVAX–based VAXELN farms. This work was suported in part by the U.S. Department of Energy under contract DE–AC02–76ER03130.A022–TC.

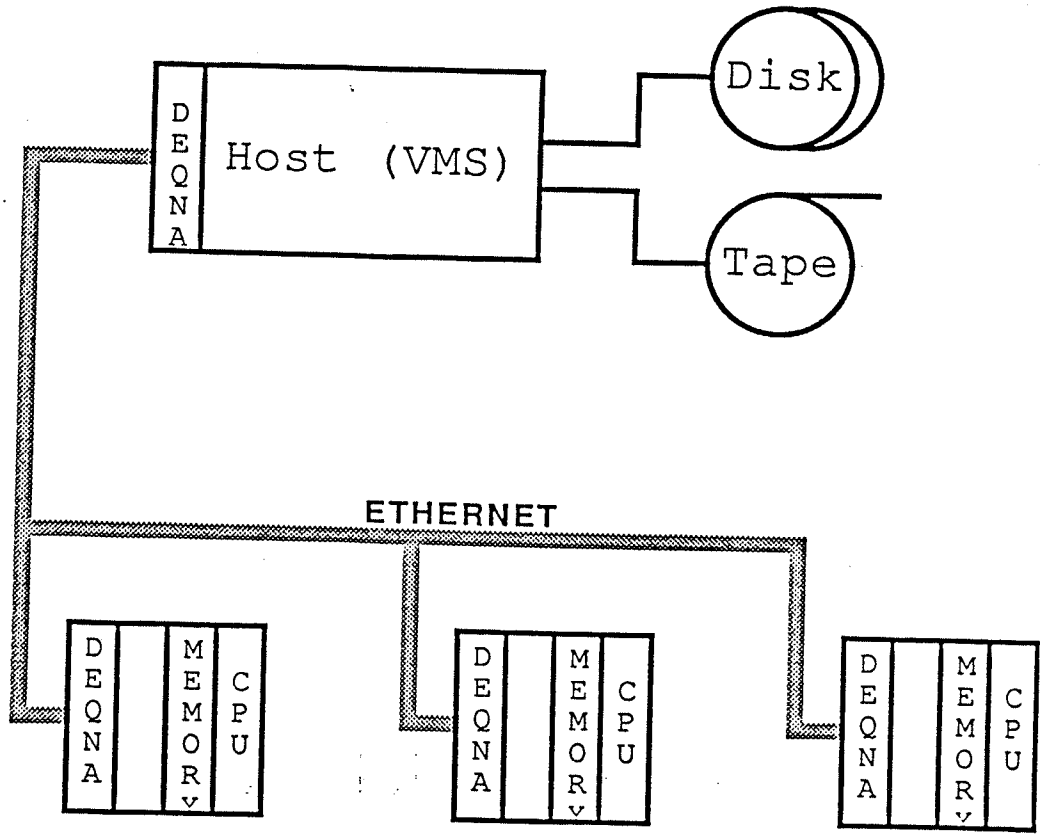
References

[1] "The MicroVAX Based Data Acquisition System for DØ", D. Cutts et al. Proceedings of the Fifth Conference on Real–Time Applications in Nuclear, Particle and Plasma Physics, IEEE Transactions on Nuclear Science, Vol. NS–34 (August 1987).

[2] "Simulating DØ and Hermiticity Studies", A.M. Jonckheere, Presented at the ANL–Monte Carlo Workshop, August 1987 (to be published).

[3] Zeller Research Ltd., 8 Rushton Drive, Cranston, RI 02905

Hardware Layout of Farm



VAXELN nodes.
'Infinite' number may be added.

ETHERNET/DEQNA communication may be replaced by high speed I/O bus as in DØ data aquisition system.

Figure 1: The basic hardware setup for the GEANT run.

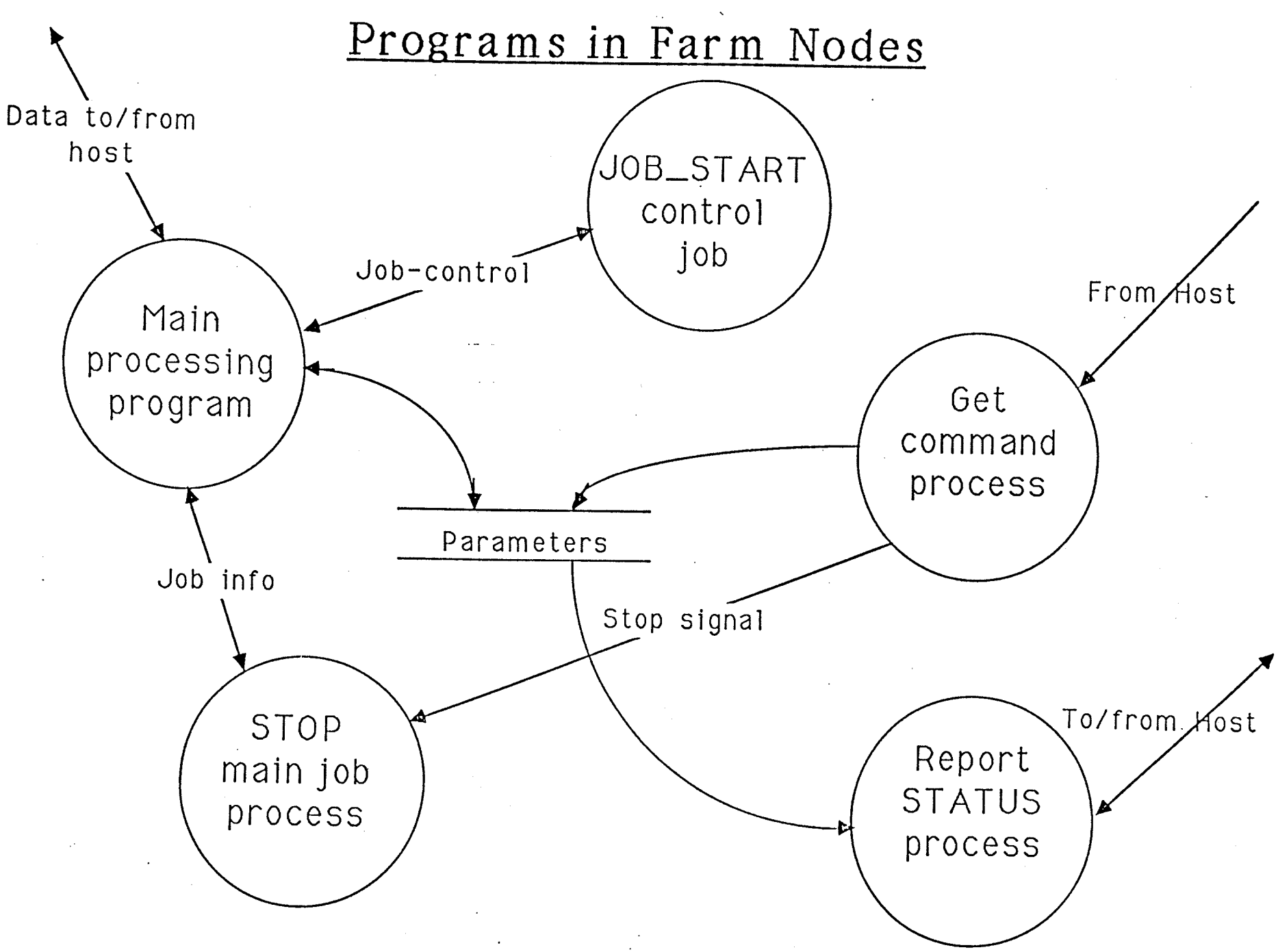


Figure 2: Schematic diagram of the programs running in each farm node.

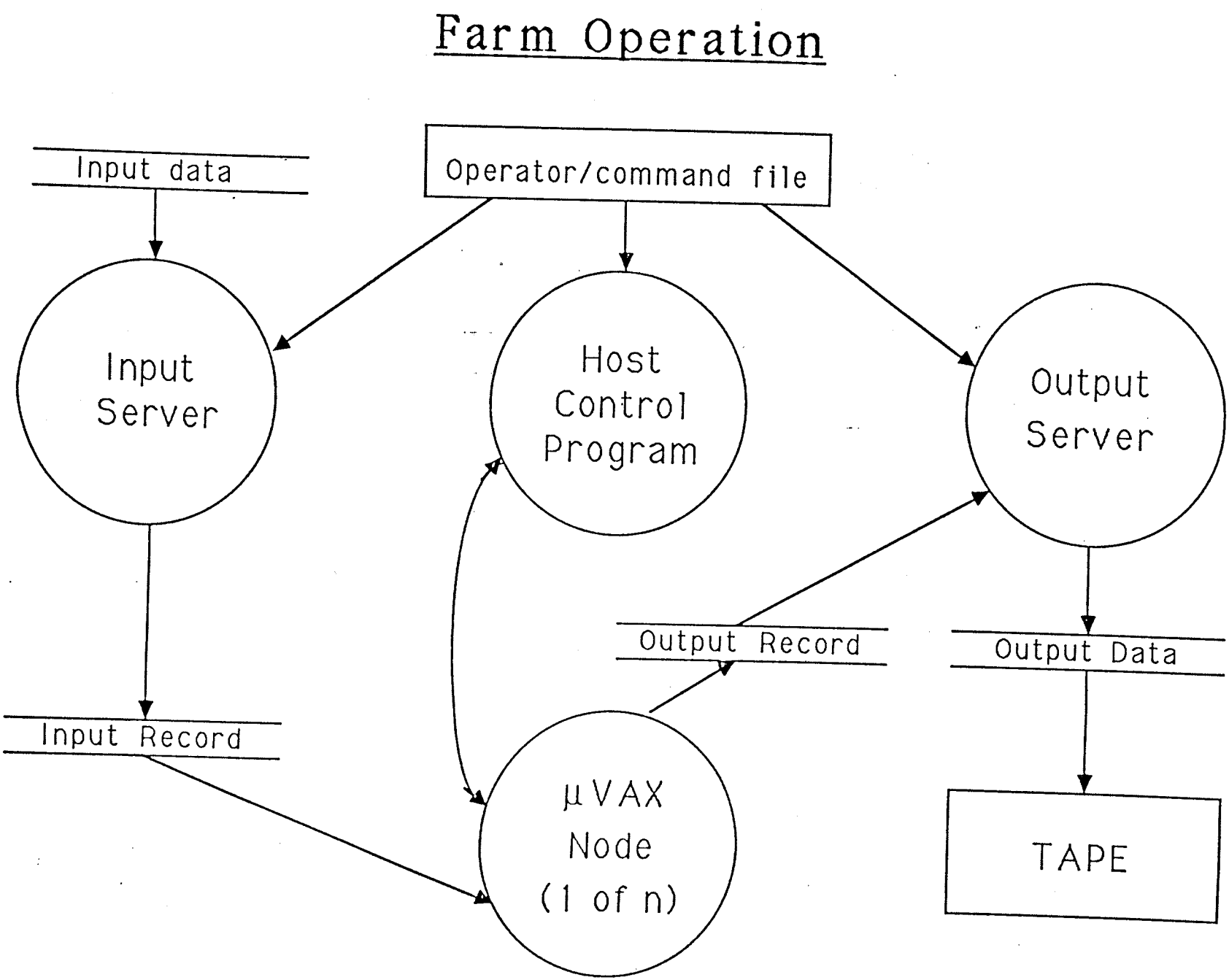


Figure 3: Schematic diagram of how the farm operation occurs.