High performance I/O in supercomputing applications

V. Zecca

IBM ECSEC European Center for Scientific and Engineering Computing, Viale Oceano Pacifico 173, 00144 Rome, Italy

ABSTRACT

At the IBM European Center for Scientific and Engineering Computing a prototype has been written for transparent transformation of I/O primitives into access to processor storage, thus allowing applications to continue to use standard high level language I/O statements, but taking advantage of the full storage hierarchy provided by the underlying processor in a VM/ESA environment. The prototype for High Performance CMS I/O (HPCMSIO) may be used with high level programming languages like FORTRAN, C, PL/1, and Assembler, of course. Purpose of the work was to demonstrate the capability of VM/ESA to optimally execute an industrial code on a supercomputer exploiting its storage hierarchy, and particularly the virtual and expanded storage, with negligible intervention on the programmer's side.

The present paper discusses the implementation of these concepts and provides performance data measured with the fluid dynamics industrial code FIDAP (*). For I/O to memory a speed of 60 Megabytes/sec was measured on an ES/9000 model 620. Such a speed is measured when the data space resides entirely in central storage. Otherwise, page faults occur and pages are moved between central storage and slower access media like Expanded Storage or disk. In any case data space access is always faster than application disk access because of the shorter access time for Expanded Storage and of the faster paging as done by VM/ESA. For a machine with a shorter cycle time like the ES/9000 model 900 a speed of 100 Megabytes/sec can be achieved for I/O to memory.
(*) FIDAP is a registered trademark of Fluid Dynamics International.

INTRODUCTION

Continuing advances in storage technologies, like the introduction of very large addressing schemes and of block-addressable processor storage as the IBM expanded storage, suggest that better performances can be obtained with supercomputers like the IBM ES/9000, if these codes are optimized in their I/O handling. This is particularly true for solvers originally written for machines provided with a small amount of central storage. It is now possible to take advantage of new storage technology and to move data from I/O devices to storage with no modification to the applications.

The constant increase in performance and reliability of modern mainframes allows the solution of larger problems in several areas such as engineering, physics, and more generally in scientific and technical computing. Problems that were insolvable a few years ago are now handled efficiently by computers such as the IBM ES/9000. In the past years we have seen a consistent upgrade of computer architectures to enable execution of industrial and academic applications requiring a larger range of addressing.

The IBM System/370 Extended Architecture (XA) increased the address range on IBM mainframes by a factor of 128 over the earlier IBM System/370 architecture. Today, we are beginning to feel a need to go beyond the two gigabytes addressing scheme of XA for very large applications in fluid dynamics, aerodynamics, and similar areas, where solutions for problems like the flow around an airplane, or a space shuttle, are sought. This is a predictable evolution from earlier studies concerning local flow around an airfoil or inside a nozzle, that were fully contained within the S/370-XA addressability limitations.

The IBM Enterprise Systems Architecture (ESA/390) [1] provides a virtually unlimited number of two-gigabyte address spaces to user applications in a VM/ESA 1.1 environment through VM data spaces [2]. We wrote a prototype to exploit the extensive addressing capability of VM/ESA in a high level programming language environment. While polishing this paper, on May 20, IBM announced VM/ESA 2.1. VM/ESA Release 2.1 delivers the virtual disks in storage support. This high-speed, data-in-memory capability can provide performance improvement in external throughput for applications similar to what is provided by the prototype. However, no performance data is available at this time.

In engineering applications, if the storage available is not sufficient for solving the problem in-core, the solver typically splits the problem matrix into blocks that are put in, say, a FORTRAN logical unit. For very complex geometries, the limit of two Gigabytes for a single address space in IBM System/370-XA can well be surpassed. In that case, heavy I/O activity can reduce performance of the solver. VM/ESA permits fast access to data spaces and solvers may take advantage of that feature by transforming access to the spill unit into access to a data space.

The intent of this paper is to show how data spaces were put to work with FORTRAN codes. The findings and the experience gained are detailed in the following sections. The following section describes in more detail the ESA/390 architecture.

EVOLUTION TO ESA/390

Enterprise System Architecture/390 (ESA/390) is the next step in the evolution of the IBM architecture that started with System/360, continued with System/370, and was extended with 370-XA and ESA/370. ESA/370 includes all of the facilities of 370-XA and offers major advanced address space facilities that further increase the amount of virtual storage available for programs. ESA/390 includes ESA/370 and significant additional addressing facilities.

The 370-XA architecture extends System/370 architecture and gives application size constraint relief. For our purposes the major extension is bimodal addressing. Bimodal addressing provides a 24-bit addressing mode for the execution of older System/370 programs and a new 31-bit addressing mode, thus extending the virtual address space size available to a program to two gigabytes.

ESA/370 is built on top of 370-XA and offers advanced address space facilities that greatly expand the amount of virtual storage available to programs and permit programs and data to reside in different address spaces. In particular ESA/370 includes hardware addressing facilities that let programs access multiple address spaces simultaneously, extending the previous 2GB addressing limit.

ESA/390 is an extension of ESA/370 that offers several new facilities, in particular the capability for VM/ESA to provide VM data spaces. Using this extension, VM/ESA provides a virtual machine architecture called XC (Extended Configuration) that has capabilities for extended addressing and data sharing for application programs. XC is for virtual machines only and is especially designed for CMS (Conversational Management System) applications and helps user programs to concentrate on the problem, letting VM/ESA to manage address spaces, thus relieving much of the burden for address space management from the application program, which needs to keep track only of the names of the address spaces it uses. Thus a virtual machine in XC mode can address its primary address space, that may containing programs and data, and a virtually unlimited number of additional address spaces which may contain only data, which for this reason are called data spaces.

DATA SPACES

The guest main storage of an XC virtual machine consists of one or more extents of storage known as address spaces. At logon, each virtual machine has an initial extent of main storage, called its primary address space. An application running in an XC virtual machine can create multiple address spaces of up to 2GB each for data storage in addition to its primary address space. These address spaces are also called data spaces because they can be used only for data storage and manipulation.

Large industrial applications require vast amounts of storage to work efficiently. For example, the graphic representation of three-dimensional objects, fluid dynamics programs, and image processing all require such large storage areas. Current support for data spaces is given by IBM for VS FORTRAN COMMONs that can be mapped onto data spaces. This allows an application to have multiple huge COMMONs simultaneously. In this paper it is described another exploitation of data spaces, namely mapping I/O units onto data spaces. While exploiting data spaces for COMMONs require recompilation of the source programs, no recompilation is needed

with the High Performance CMS I/O (HPCMSIO) prototype developed at the IBM European Center for Scientific and Engineering Computing (ECSEC). With HPCMSIO the user modifies only the execution procedure for the application by adding the HPCMSIO call.

THE HIGH PERFORMANCE I/O PROTOTYPE

The High Performance CMS I/O software prototype promotes flexible exploitation of the supercomputing capabilities of the ES/9000 computers with the S/390 architecture for a high level language (HLL) as, for example, FORTRAN. Such an exploitation has been designed specifically for transparent high performance I/O in a CMS virtual machine with the VM/ESA Operating System. Taking FORTRAN as an example of HLL, FORTRAN files are mapped onto VM data spaces thus achieving an I/O data rate of 60 Megabytes/second measured on an ES/9000 model 620 for a filesize of 40 Megabytes. For the same test application run with I/O on a 3390 disk the measured I/O speed was of 1.4 Megabytes/sec. HPCMSIO works with VM/ESA 1.1 and later releases. Both sequential and direct access files are supported. Maximum extension for a file is two gigabytes. Another case where it could be useful to exploit data spaces is when it is not available enough contiguous disk space to run an application. Disk space can then be allocated on one or more data spaces with no need for a request to the system administrator.

No modification at all on the FORTRAN source programs is required. No recompilation or relinking/regeneration is needed: HPCMSIO can just work on executable (CMS or load) modules. The only modification needed is in the execution procedure for the FORTRAN program that needs invocation of the HPCMSIO services. Data Spaces are volatile, however, because they are dropped by VM/ESA at logoff time. Therefore, only scratch data should be put on data spaces.

Invocation of HPCMSIO

HPCMSIO is first loaded as a nucleus extension with the following CMS command:

```
NUCXLOAD HPCMSIO ( SERVICE
```

and then every high performance file has to be defined with the following command:

```
HPCMSIO VIO filename filetype filemode extent
```

where VIO denotes virtual I/O onto a data space. For VIO the extent clause informs HPCMSIO about the extension of the data space which is given in Megabytes. Up to two gigabytes of data space size are supported for the VIO capability. Multiple VIO units are supported.

IMPLEMENTATION DESCRIPTION FOR HPCMSIO

The designed environment for HPCMSIO is CMS with VM/ESA 1.1 or later. The code comprises the HPCMSIO module, a CMS Nucleus Extension that intercepts I/O calls to CMS and transforms them into data space access. No modification to either VM/ESA, CMS, or the HLL environment is needed. HPCMSIO only acts on files defined as VIO. Any other file is handled with the standard CMS/HLL protocol.

HPCMSIO Internals

HLLs like VS FORTRAN run time libraries use OS macros for I/O. These primitives are intercepted by CMS and transformed into CMS native I/O primitives. The CMS native I/O primitives are then intercepted by HPCMSIO with a software technique known as PSW stealing and substituted with access to data spaces.

PERFORMANCE RESULTS

The hardware configuration comprised an IBM ES/9000 model 620 with 64 Megabytes of Central Storage and 384 Megabytes of Expanded Storage. For testing of the prototype an industrial application was selected. The fluid dynamics FIDAP Finite Element code [3], that analyses 2-D and 3-D incompressible viscous fluid flow, was chosen. The FIDAP fluid dynamics package, along with similar packages such as ADINA-F, is a general purpose computer program that uses the Finite Element Method (FEM) to simulate many classes of viscous incompressible fluid flows. Two dimensional axi-symmetric, and 3-dimensional steady state or transient simulations in complex geometries, including the effects of temperature, are possible. The analysis is limited only by practical considerations of computer time and the capacity of auxiliary storage devices for problems that cannot be solved in core. Versions of FIDAP exist for a wide range of computers under many operating systems.

In FIDAP, if the storage available is not sufficient for solving the problem in-core, the solver splits the system into blocks that are put in an external data set. For very complex geometries, the limit of two Gigabyte virtual addressing in System/370-XA can well be surpassed. In that case, heavy I/O activity can reduce performance of the FIDAP solver. With the High Performance I/O prototype FIDAP took advantage of VM data spaces by transforming access to the spill data set into access to a data space. A test case from the test case suite for FIDAP was selected [4], and it was run with standard I/O to disk and with the prototype. This case had a mesh with 71×71 points, with 18,378 equations, 16,028,372 elements, and a mean half-bandwidth of 436. An area of 120 Megabytes in central storage was allocated for FIDAP buffers, and three blocks with 60 Megabytes each were required to spill the matrix to be solved onto auxiliary storage.

The table summarizes the performance results of the original (I/O) version and the enhanced (VIO) version. Times are given in minutes. With data space exploitation all the I/O done to the spill unit is avoided, thus greatly decreasing the total elapsed time. A comparison is then done with a large buffer space such that no spill unit is needed (Storage).

Table 1. Performance of FIDAP in VM/ESA

Version	CPU Time	Elapsed Time
I/O	7.78	32.55
VIO	8.35	15.72
Storage	7.17	12.42

Notice how CPU time is nearly independent from the I/O method employed. With VIO some CPU time is spent in transferring data to/from data spaces, but this excess time is more than gained with a much shorter elapsed time.

CONCLUSIONS

The IBM Enterprise Systems Architecture/390 (ESA/390) incorporates unique features that are of benefit in Numerically Intensive Computing (NIC) applications. Features like XC (Extended Configuration) provide data spaces, which are available to high level languages, like FORTRAN, to enable NIC applications to take advantage of the full power of the Enterprise Systems Architecture, thus providing an I/O speed of 60 Megabytes/sec on an ES/9000 model 620 in a FORTRAN program with no source modifications.

BIBLIOGRAPHY

[1] **Special issue on Large Systems,** IBM Systems Journal 28 (1989).

[2] **Special issue on VM/ESA,** IBM Systems Journal 30 (1991).

[3] **FIDAP General Introduction Manual,** Fluid Dynamics International, September 1987.

[4] V. Zecca, **Exploitation of Data in Memory in the FIDAP code on the IBM ES/3090,** ASE 91 Proceedings, 1991.