Design of the IBM System/390 computer family for numerically intensive applications: An overview for engineers and scientists

by D. H. Gibson
G. S. Rao

The IBM System/390® (S/390) computer family provides a two-order-of-magnitude performance range for numerically intensive applications. The engineer or scientist can use the same operating system, compiler, and run-time environment commonly across the family. This paper provides an overview of primary S/390 hardware and software products of interest for numerically intensive applications, including MVS/ESA™, VM/ESA®, AIX/ESA™, and the extension of FORTRAN for very large applications and parallel applications. The primary portion of the paper is focused on details of design interest in three specific hardware products within the S/390 family, with emphasis on the Enterprise System/9000™ (ES/9000™) Model 900. Also described is a potential parallel-computing configuration using the ESCON Director™. The paper concludes with a discussion of the generic system environments within which S/390 products can support the technical user.

1. Introduction

Engineers and scientists have historically shown interest in the design details of the tools they use, especially computers. The history and a general description of computers as a tool for engineers and scientists have been given in an earlier paper by Gibson et al. [1]. The present paper provides design details and references of interest for users of System/390® (S/390) in numerically intensive applications.

The paper begins with an overview of the elements of an S/390 system that are of interest to an engineer or scientist. This is followed by a survey of syntactical

extensions of FORTRAN for very large applications and parallel processing, including the preprocessor and the related syntax of the supercomputing systems extension[1] Clustered FORTRAN, each of which represents pioneering efforts in coarse-grained parallelism. The MVS/ESA™ and AIX/ESA™ operating systems are discussed next, followed by an extended discussion of design details of interest in three models in the Enterprise System/9000™ (ES/9000™) family, with special emphasis on the Model 900. This fastest ES/9000 model within the S/390 family provides scalar, vector, and parallel computing in the traditional manner of improving state-of-the-art technology and system structures. The advances in semiconductor technology, packaging technology, and clock speed are complemented by multiple execution elements and a storage hierarchy that includes a two-level cache design. The vector facility for the Model 900 is also discussed, followed by a description of a potential base for parallel processing in the S/390 environment using the ESCON Director™. The ESCON Director is a part of the Enterprise Systems Connection (ESCON™) family of products that improve data rate, availability, distance, connectivity characteristics, and system management for system interconnection. It is a nonblocking crosspoint switch with up to 60 ports. The paper concludes with a discussion of the system usage environments of interest to engineers and scientists for the ES/9000.

2. S/390 overview from an engineering and scientific perspective

The S/390 computing subsystems of interest to an engineer or scientist are the high-level languages, the operating systems, the processors, the storage subsystems, and the hardware connectivity. S/390 computing services may be invoked from UNIX® workstations, from OS/2® or DOS workstations, or from a variety of mainframe-interactive (MFI) terminals.

• *High-level languages*

Numerically intensive computing usually involves calculations written in a high-level language. The predominant language for engineering and scientific numerical computing is FORTRAN. Section 3 of this paper discusses selected FORTRAN capabilities of interest within the S/390 family, including the preprocessor for generation of in-line subroutine code and the parallel extensions known as Clustered FORTRAN.

S/390 computing services may also be invoked using high-level languages other than FORTRAN, which have their own special merits. Subroutines written in any of the following high-level languages can call and be called by FORTRAN: The C language is often used in applications written for a UNIX environment. C/370 [2] provides C language support for large memory and multitasking. The APL language has long been regarded as providing an exceptional ability to formulate a task in array and vector constructs. APL2™ [3] automatically uses the Vector Facility of S/390 (when present) for improved performance. VS PASCAL [4] embodies many software engineering concepts and provides IBM extensions to the ANSI and ISO PASCAL standards. Assembler support is provided by the Assembler H [5] product.

In addition to high-level languages, a library of often-used routines that have been highly tuned to the ES/9000 model is available. McComb and Schmidt [6] describe the contents and some of the techniques used in ESSL, the Engineering and Scientific Subroutine Library. This library comprises more than 280 routines in ten computational areas including, for example, linear algebra, matrix operations, signal processing, and sorting. It has been demonstrated to achieve the fastest possible execution rates on ES/9000 models.

Another library of interest is the Optimization Subroutine Library (OSL), a collection of high-performance mathematical subroutines for use in application programs that solve optimization problems. The mathematical programming techniques within OSL, which seek to minimize or maximize a function subject to a set of constraints, include linear programming (simplex and interior point methods) [7–10], mixed-integer programming [11], quadratic programming [12], and pure network solution [13].

MPSX/370 (Mathematical Programming System Extended/370) Version 2 is a program used to optimize operations and investments common to most users. MPSX provides solutions for linear and separable mathematical programming problems and, with Version 2, supports the Vector Facility for improved performance.

Several hundred application programs [14] in the public domain or available from vendors have been enabled to support the Vector Facility, in addition to the thousands that are enabled for the scalar ES/9000 processors. Examples of the vector-enabled applications are MSC/NASTRAN® for structural analysis, the FLO codes for computational fluid dynamics, the GAUSSIAN codes for chemistry, and the ECLIPSE codes for seismic processing. In addition there are codes for physics, for electronics, and for quantitative analysis.

• *Operating systems*

IBM System/390 services are available to the engineer or scientist in one of three possible operating system environments[2]. The UNIX environment is provided by the

[1] SuperComputing Systems Extensions (SCSE) are additions to the S/390 family. They are designed to enhance S/390 for supercomputing. A specific SCSE, such as Clustered FORTRAN, is available via special agreement.

[2] There is a fourth operating system, VSE/ESA™, available for S/390 that is not described herein because of the low use of this operating system by engineers and scientists.

AIX/ESA native operating system and by the AIX/370™ operating system, which takes advantage of the paging and other hardware resource services provided by VM/ESA®. Many users prefer the familiar interactive environment offered by CMS under VM/ESA. Production environments such as CATIA®-based simulations often choose MVS/ESA and the associated batch monitor Job Entry Subsystem, as do users concerned with systems storage management.

- *Processors*

The eighteen ES/9000 processors announced in September 1990, together with the additional seven processors announced in September 1991, span a 100-fold performance range without vector processing, and a 200-fold range with vectors. They are based on one of three implementations, each with its own technology, logical design, and associated packaging: water-cooled (with a tenfold performance range), frame-mounted air-cooled (with a fourfold performance range), and rack-mounted air-cooled (with a threefold performance range). Section 5 of this paper details the fastest processor within each of the three implementations, with emphasis on numerically intensive computing capability.

- *Hardware connectivity*

IBM System/390 connectivity is based on the IBM Enterprise Systems Connection (ESCON) Architecture™. Five papers in this issue are devoted to ESCON. One especially interesting capability, the crosspoint switch, is described by Georgiou et al. [15]. Section 5 of this paper treats another aspect of ESCON that has not been widely discussed: the potential for an n-way parallel system based on ESCON, where $n \leq 60$. Additionally, and of particular interest to the engineer and scientist, is the FDDI (Fiber Distributed Data Interface, a high-speed fiber optic LAN) attachment to S/390 as described by Coleman et al. [16] in this issue.

3. FORTRAN, the language environment most used for NIC

- *FORTRAN 2.5*

IBM VS FORTRAN [17] Version 2 Release 5 consists of a compiler, an interactive debugging facility, and an execution-time library of subprograms. The FORTRAN 77 standard constructs are supplemented with IBM extensions. FORTRAN 66 constructs are also supported. Both scalar and vector run-time environments are supported.

One class of IBM extensions is referred to as Extended Common. Currently, an application program may be limited by the size of the common area. With this extension, the size of the application program may be very large, and recent measurements show that applications as large as 5 GB can be run efficiently. Further detail is provided in Section 4 of this paper.

Extended Common is a compiler option, not a syntactical language extension; it allows the FORTRAN programmer to define multiple extended common blocks, each as large as 2 GB in size. This is achieved by dynamically allocating storage to FORTRAN-named common areas from Enterprise Systems Architecture/390™ (ESA/390™) data spaces [18]. Neither a FORTRAN array nor a named common area may be assigned to more than one data space. The size of the application program, however, may be very large, since even with the 2GB limit on an array or named common area, the user is free to organize many arrays and variables into many named common areas. Extended Common cannot be used with static or interactive debug, the multitasking facility, or other high-level languages. Assembler language programs must be modified to accept data passed from an Extended Common block. Otherwise, the programmer continues to use normal application design practices and coding styles. The above restrictions and details of use of Extended Common are found in [17].

In order to assign named common areas to ESA/390 data spaces, the user has only to specify the compiler option "EC," followed by the list of common area names, in the same manner as for the FORTRAN dynamic common "DC" option used with System/370™ Extended Architecture (XA). In fact, if the aggregate size of FORTRAN dynamic commons in existing code grows beyond the System/370-XA limit, existing dynamic common specifications may be changed to Extended Common specifications by merely changing the option "DC" to "EC." The FORTRAN compiler organizes the named commons in the "EC" specification among a minimum number of ESA/390 data spaces and establishes all required addressability. A subroutine which explicitly references data in Extended Common is said to be running in extended mode (EMODE). A subroutine which accepts arguments that might reside in Extended Common must also run in EMODE, and must be compiled with the "EMODE" compiler option to permit it to accept Extended Common data. It is recommended when using Extended Common that optimization level OPT(2) or above be selected for compilation to avoid the greater CPU utilization experienced at lower optimization levels.

Another class of IBM extensions is referred to as Parallel FORTRAN. Parallel FORTRAN provides for automatic parallel execution of eligible DO loops and automatic integration of parallel and vector processing on a single, virtual, multiprocessing computer running under the control of either the MVS/ESA or the VM/ESA operating system.

Although automatic detection of parallelism may be an easy way to introduce parallelism, it does have some limitations [19]. For this reason, Parallel FORTRAN provides language extensions with which the programmer may specify parallel execution. These include extensions for parallel loop iterations, parallel statement sequences, and parallel subroutine execution.

Library routines for synchronizing parallel pieces of work are also provided. In-line language extensions, such as PARALLEL LOOP, allow the programmer to identify loops or blocks of statements that can be executed concurrently. Out-of-line language extensions, such as SCHEDULE TASK, permit the programmer to create asynchronous execution environments. The combination of Parallel FORTRAN and ES/9000 multiprocessors such as the Model 900 can provide a significant reduction in turnaround time for applications.

Parallel FORTRAN function is extended by two supercomputing system extensions referred to as Clustered FORTRAN.

• *Clustered FORTRAN*

IBM Clustered FORTRAN [20] and IBM Enhanced Clustered FORTRAN are combinations of hardware and software which allow IBM multiprocessor systems to be connected as clusters with all resources devoted to a single FORTRAN application. The hardware consists of high-speed intersystem connections. The two differ primarily in the hardware cluster that is supported. IBM Clustered FORTRAN allows two Enterprise System/3090™ (ES/3090™) multiprocessor systems to be connected as a cluster, whereas IBM Enhanced Clustered FORTRAN allows up to four ES/9000 multiprocessor systems to be clustered. The FORTRAN compiler and library provided as part of IBM Clustered FORTRAN are used for writing and executing the job. One FORTRAN job can execute in parallel across all of the processors, up to the maximum cluster size of twelve 3090™ central processors (CPs) or twenty-four ES/9000 CPs. In the description which follows, the term "Clustered FORTRAN" is used for software concepts which are common to both IBM Clustered FORTRAN and IBM Enhanced Clustered FORTRAN.

Clustered FORTRAN extends the Parallel FORTRAN function: Whereas Parallel FORTRAN is limited to a single operating system environment, the Clustered FORTRAN execution environment supports multiple multiprocessors each running a copy of the preferred operating system. Either the MVS/ESA or the VM/ESA operating system may be chosen, but both cannot be used in the same cluster. The extensions over Parallel FORTRAN include

- Operating system support of the high-speed hardware connection.
- FORTRAN run-time virtual configurability.
- Parallel scheduling constructs across multiple virtual computers.
- Enhanced parallel language constructs.

A user of Clustered FORTRAN specifies a virtual configuration which suits the application at hand. A real cluster configuration consisting of two six-way ES/9000 multiprocessor systems might be thought of as twelve uniprocessor computers, as four computers each with three processors, as two computers each with six processors, or many other configurations. The virtual configuration is specified at execution time by a file containing COMPUTER statements, with one statement for each virtual computer desired. When a Clustered FORTRAN application is submitted for execution, the control program extensions create the virtual configuration and map it onto the real configuration for the duration of the job.

A Clustered FORTRAN application sees a virtual configuration for its execution environment. The virtual configuration consists of one or more virtual computers which are connected by a virtual connection facility. The virtual computers share no memory; this mimics the architecture of the collection of real computers in the complex. Each of the virtual computers may be a shared-memory multiprocessor; this mimics the architecture of any one of the real computers in the complex. A virtual configuration for a Clustered FORTRAN application is shown in **Figure 1**.

• *VAST-2 preprocessor*

A program offering is available for preprocessing FORTRAN programs to optimize them where possible. VAST-2® for VS FORTRAN is a software tool that automatically optimizes VS FORTRAN programs by preprocessing DO and IF loops. It improves performance through parallelization, vectorization with enhancements, and in-line expansion [21].

Parallelization detects when the iteration of a loop can be spread across separate processors; vectorization enhancement inserts compiler directives, restructures loops, and substitutes calls to library routines, in order to increase the number of loops vectorized, and in-line expansion reduces calling overhead and assists other kinds of optimization by inserting the bodies of called routines at the places where they are called.

IBM VS FORTRAN does a large amount of optimization on its own. VAST-2 is intended for use on programs that require the maximum possible optimization. As a preprocessor, it is able to do some higher-level optimizations that would be inefficient to attempt in the compiler itself. VAST-2 also converts FORTRAN 8X array syntax into DO loops that can be processed by the VS FORTRAN compiler.

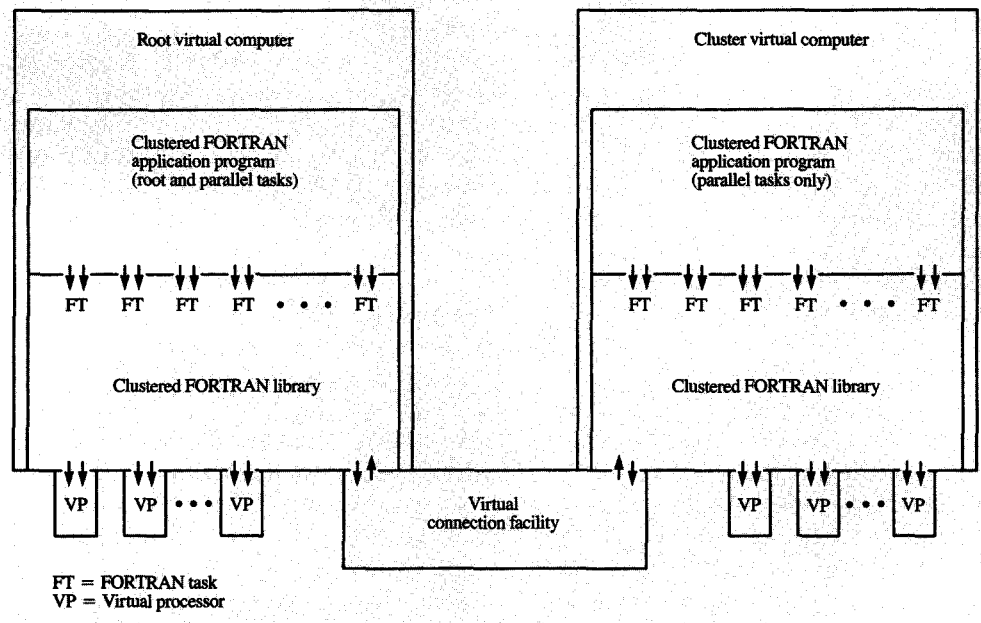

Figure 1

A virtual configuration for a Clustered FORTRAN application.

4. AIX, VM/ESA, and MVS/ESA operating system environments

• *AIX/ESA*

Many engineers and scientists want to transport applications among systems without the necessity of rewriting for each system. High-level languages (HLLs) are helpful if each system has a compiler that maps the chosen HLL to the specific system architecture. If it were true that no interaction between the user and the application was required, simply recompiling would suffice to transport applications among systems. However, modern-day computing practices involve the user working at a terminal to interact with the application. The interaction always involves set-up, often involves redirection of processing, and sometimes involves debugging. The operating system determines the environment that the user must understand for this interaction, and UNIX is the only operating system that attempts to run on architecturally different systems. If there is no UNIX on the intended system, the user must learn a different operating system.

The native version of UNIX for ES/9000 processors is AIX/ESA, which became available in the second half of 1992. AIX/ESA is the newest member of the AIX® family, providing a native implementation of the OSF/1™ operating system for IBM Enterprise Systems. In support of the heavy CPU and storage requirements of numerically intensive applications, AIX/ESA exploits hardware features such as multiprocessing and large-process address spaces (up to 2 GB). AIX/ESA also supports expanded storage directly as a paging device, providing fast paging performance for large processes.

AIX/ESA operates natively, or under the VM/ESA operating system, or in a logical partition of an ES/9000 processor. Migration and compatibility features facilitate ease of migration from the current UNIX offering, Advanced Interactive Executive/370™ (AIX/370™).

• *AIX/370*

The UNIX environment is currently supported by the AIX/370 operating system under VM/ESA.

While there has been concern expressed about the performance of running UNIX production under VM, analysis has shown that there is no visible performance degradation due to running a suitably configured AIX/370 guest under VM/ESA. This is primarily because VM employs advanced paging algorithms, and it is particularly advantageous to use VM paging when a large address space is required that exceeds the physical memory of the machine. Large address spaces are typical of numerically

intensive computing. Blandy and Newson [22] describe the management of host storage paging using algorithms that are carried forward into VM/ESA. In particular, Koba and Iimura [23] state that "when an AIX/370 guest has been set up with a processor dedicated to it, all DASD devices dedicated to it, multiple swap devices allocated to it, and a large pool of reserved pages in memory, CPU intensive applications will run at essentially the same speed or better as under CMS. This set up has eliminated much of the non-native overhead associated with CP while taking advantage of the superior VM paging subsystem and the microcode assists available."

- *VM/ESA*

The VM/ESA operating system provides VM data spaces through a virtual machine architecture called Enterprise Systems Architecture/Extended Configuration (ESA/XC). Gdaniec and Hennessy [24] describe the ESA/XC virtual machine architecture and present an overview of the VM/ESA services provided in support of VM data spaces.

For engineers and scientists, one of the more interesting services is the Shared File System (SFS). SFS provides a hierarchical view of files that may be shared among the several thousand users common to a large VM/ESA system. The fundamental performance problem of a reentrant service supporting thousands of users in a singular limited storage space is eliminated by multiple data spaces.

- *MVS/ESA*

The MVS operating system provides a number of capabilities of interest to the engineer or scientist. Among these capabilities is a rich batch monitor system and system-managed storage. One especially interesting capability in S/390 is a callable system service whereby the user can give hints to the operating system that result in a faster-running application. This is known as reference pattern services.

Long-running production applications, such as CATIA-based simulations, often reference large amounts of virtual storage in a repeatable, predictable manner. MVS/ESA SP 4.2, the S/390 version and initial release of MVS, provides reference pattern services through program CALLs. Judicious use of these CALLs benefits run-time performance. In addition, when the MVS/ESA operating system can detect forward-sequential accessing of virtual storage, the CALLs need not appear in the application to obtain the performance improvement.

The numeric data processed by such applications often span many of the 4096-byte pages that are managed by MVS. The basic design of MVS is oriented to demand paging, meaning that one page is brought into central storage each time a page fault is taken. The page that is brought in is the one containing the data demanded by the application. Reference pattern services allow the operating system to bring multiple pages into real storage when a single page fault is taken. An application can provide page-reference pattern information that identifies the range of data being referenced, the reference pattern, and the number of bytes that the system is requested to bring into central storage at one time. This information is provided in a program CALL, the details of which are described in an IBM publication on callable system services [25].

- *Emerging application example*

To demonstrate the kind of problem handling that reference pattern services make practical, a large aircraft flow-field analysis program has been measured. The grid has been refined such that a 5GB problem size is created. The FORTRAN Extended Common capability described earlier in this paper has been used to define and dimension the necessary variables. Thus, the system as a whole contributes to the performance and coding which make it practical to handle problems of this size. The measurements are compared to a base measurement of 48 hours run time to complete two iterative cycles on an ES/3090 Model J system with 512 MB central and 2 GB expanded storage. The reference pattern services in MVS/ESA 4.2 allow this job to run in 8.5 hours. The ES/9000 Model 900 (see the next section of this paper) allows this job to run in about 6.9 hours; the CPU time is reduced by a factor of 2.4, but the I/O wait time is not reduced. Note that in a production environment where the system is not dedicated to running this one job, the improvement in processor speed affects a larger fraction of the total turnaround time. Increasing the processor storage capacity of the ES/9000 Model 900 to 1 GB of central and 4 GB of expanded storage allows this job to run in one hour. Note finally that this type of problem normally requires 100 or more iterative cycles to complete. With the ES/9000 Model 900, MVS/ESA 4.2, and FORTRAN 2.5 Extended Common, it is practical to run the complete job over a weekend.

5. ES/9000 Models 900, 480, and 170

- *Architecture overview*

ESA/390 architecture [26] provides a rich environment for numerically intensive computing applications. Addressing capabilities permit access to 2GB data spaces. Floating-point arithmetic is provided in 32-bit, 64-bit, and 128-bit formats. Primitive instructions include variations on add, subtract, multiply, divide, and square root. Logical flow is controlled with comparison and branching instructions.

On September 5, 1990, IBM announced eighteen ES/9000 processors which implement ESA/390 architecture. One year later an additional seven processors were announced. All twenty-five are compatible. They

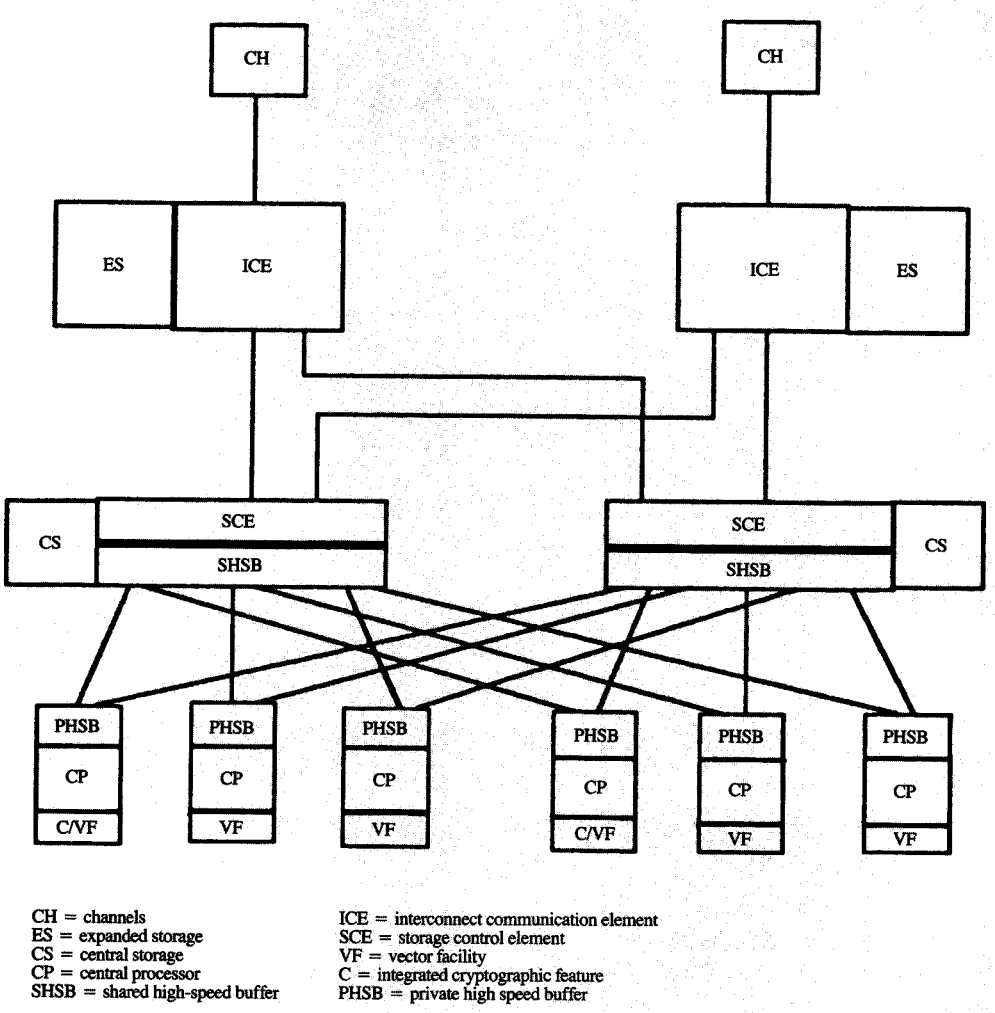

Figure 2

ES/9000 Model 900 system structure.

differ primarily in the speed of execution. For even higher performance, seventeen of the twenty-five additionally offer compatible vector operations [27].

The twenty-five ES/9000 processors may be grouped into three implementations: water-cooled; frame-mounted air-cooled; and rack-mounted air-cooled. The fastest processor within each of the three implementations is described below, with emphasis on numerically intensive computing capability.

- *ES/9000 Model 900*

System structure

Data flow The ES/9000 Model 900 is a shared-memory MIMD system with six central processors (CPs), each of which can have an optional vector facility. As shown in **Figure 2**, each CP connects to the two sides of the storage control element (SCE) and shared high-speed buffer

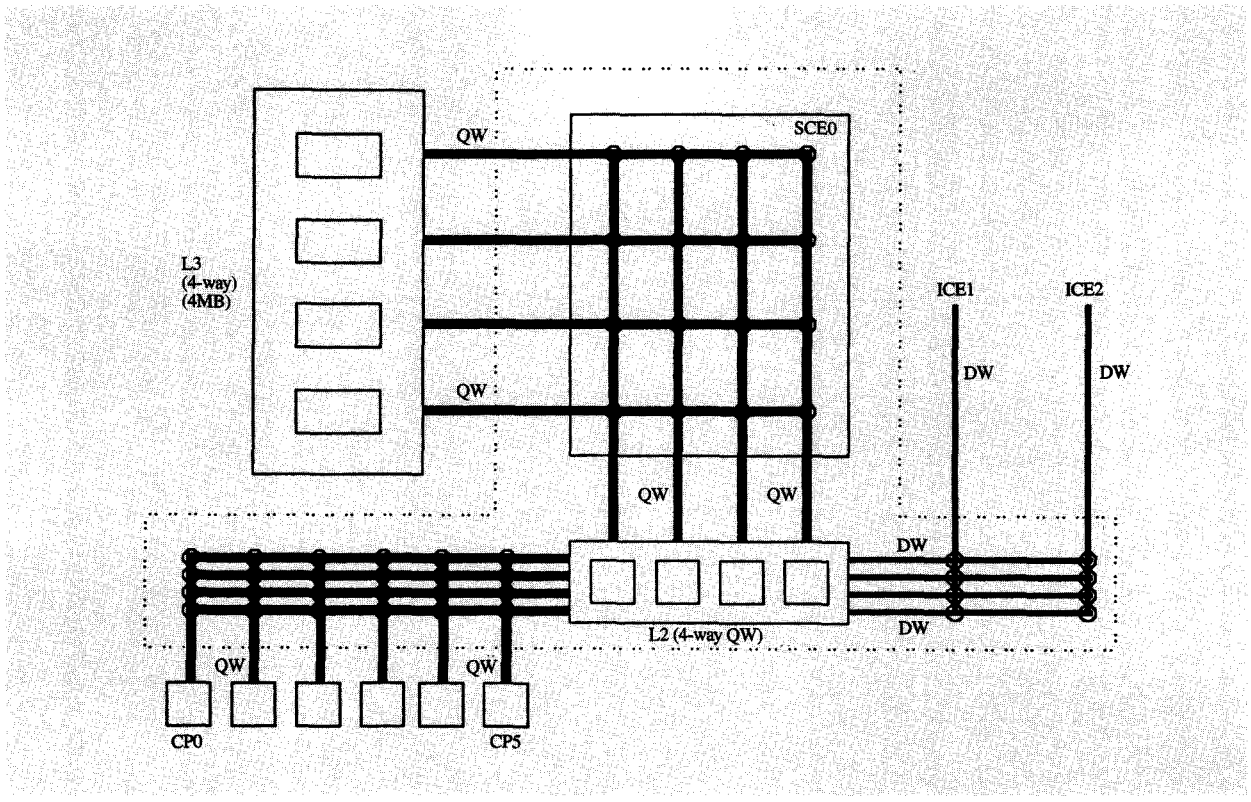

Figure 3

ES/9000 Model 900 conceptual storage control element data flow.

(SHSB) with associated controls. The connection consists of a doubleword store bus from each CP to each SCE side, and a quadword fetch bus from each SCE side to each CP. The quadword fetch bus is composed of one unidirectional doubleword bus and one bidirectional doubleword bus, described further in the subsection on the ES/9000 Model 900 Vector Facility.

The SHSB functions as a 4MB four-way set-associative store-in cache with a 256-byte line size. Each SHSB side is capable of fetching or storing four quadwords each machine cycle. Absolute addresses are mapped to the SHSB side corresponding to the 4MB mapping of central storage; a CP need only request an absolute address from one side. Requests by two or more CPs to store into the same line are resolved by the SCE using the SHSB directory. I/O activity interrogates the SHSB directory and bypasses the SHSB if there is no hit.

Each central storage (CS) side is four-way interleaved on 256-byte boundaries. It is partitioned between the two sides on 4MB boundaries. The mapping of absolute to physical 4MB ranges is determined by the processor controller.

The interconnect communication element (ICE) is connected to each SCE side by doubleword busing that supports one doubleword fetch and one doubleword store each machine cycle. The ICE supports the transfer of data to and from the expanded storage, and the transfer of data to and from channels (CH).

Thus, the data-fetch portion of each SCE side may be thought of as a 4 × 4-quadword crosspoint switch between the CS and the SHSB, cascading to a 4 × 6-quadword crosspoint switch between the SHSB and the CPs in parallel with a 4 × 2-doubleword crosspoint switch between shared SHSB and ICE. This conceptual representation is shown in **Figure 3.**

Similarly, the data-store portion of the SCE may be thought of as a 6 × 4-quadword crosspoint switch from the CPs to the SHSB. However, on any given machine cycle only one doubleword and one quadword may move between a CP and an SCE.

Memory hierarchy **Figure 4** shows details of the central processor. Each CP contains 256 KB of private high-speed buffer (PHSB), divided into a 128KB instruction buffer and

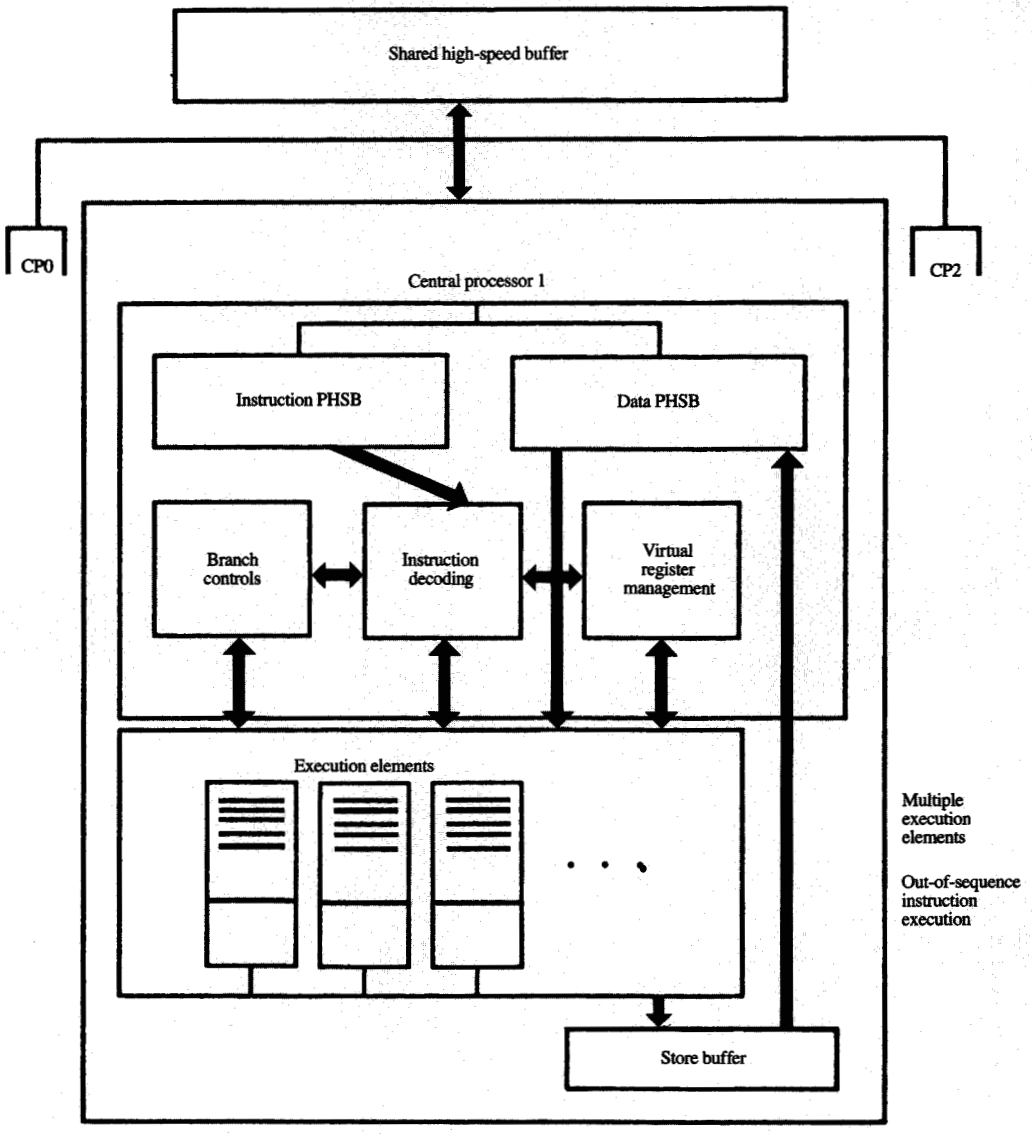

Figure 4

ES/9000 Model 900 central processor structure.

a 128KB data buffer. As described by Liptay [28], both the instruction and the data buffer are four-way set-associative store-through doubleword-interleaved, with a 128-byte line size. Access to these private high-speed buffers is overlapped with execution and is essentially transparent to the execution speed. When the required data are not contained in the data buffer, the SHSB is accessed for a 128-byte line. The delay until the first operand is available to an execution element is approximately 12 cycles when the required data are found in the SHSB. If the required data must be accessed from CS, the delay is 31 cycles. This delay is reduced when accessing consecutively higher memory addresses. Because the SHSB line size is 256 bytes, whereas the CP accesses for a 128-byte line, the

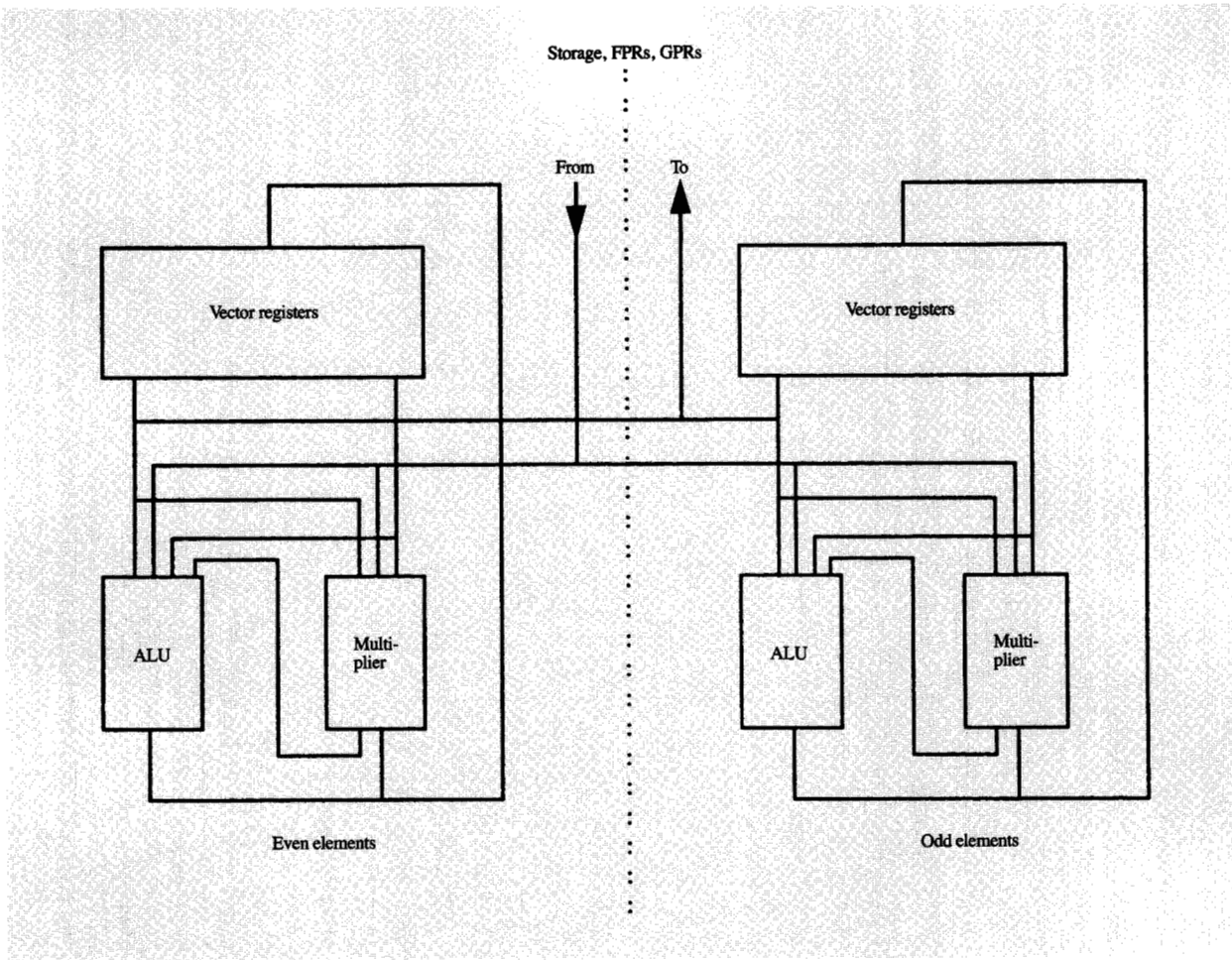

Figure 5

ES/9000 Model 900 Vector Facility, one per CP.

second half of the SHSB line is always found when accessing consecutively higher addresses; for this accessing pattern the delay averages 21.5 cycles.

For vector processing the machine has the capability to do prefetching, as described later.

Execution rates The design of the central processor (CP) is presented by Liptay [28] with emphasis on execution from the associated private high-speed buffer (PHSB). The hardware is capable of supporting the execution of three or more instructions on any given machine cycle. The floating-point execution element contains a pipelined adder and multiplier capable of producing a result each machine cycle. Measurements have shown that the LINPACK 100 × 100 benchmark achieves 30 MFLOPS on the scalar execution elements.

ES/9000 Model 900 Vector Facility

Data flow The IBM ES/9000 Model 900 Vector Facility data flow for one CP is shown in **Figure 5.**

The 8 (or 16) vector registers (VRs), each holding 256 64-bit (or 32-bit) elements, are organized as even/odd sets each holding 128 elements. Each set can deliver up to two doublewords per machine cycle to either the associated arithmetic logic unit (ALU) or the multiplier, or one doubleword to each. Simultaneously, each set can accept up to one doubleword per machine cycle. A doubleword contains either one 64-bit operand or two 32-bit operands, depending on the instruction being executed. This even/odd VR design supports the processing of four 32-bit arithmetic operations per machine cycle requiring eight 32-bit accesses and four 32-bit putaways.

An ALU and a multiplier can also be supplied via a data path that carries operands from the data PHSB on a quadword bus, or from the FPRs or GPRs, of the associated CP. Another data path into an ALU comes directly from the multiplier. This path supports the 64-bit compound vector instructions, such as multiply and add, generating as many as four 64-bit floating-point operations per machine cycle.

Upon presentation of two doublewords at the input, an ALU can produce one 64-bit result or two 32-bit normalized results two machine cycles later; a multiplier can produce one 64-bit product or two 32-bit products three machine cycles later. Both an ALU and a multiplier can accept new doublewords each machine cycle. The ALU executes add, subtract, compare, and logical operations, and is also the path for loading the vector registers. The multiplier executes multiply, divide, and square root operations. Execution of divide operations takes place at a rate of 16 quotients each 19 machine cycles once the pipeline is filled. Execution of square root operations takes place at a rate of 16 results each 22 machine cycles.

Output from the ALU or the multiplier is normally placed in the VRs. Data are sent from the VRs to storage using the bidirectional doubleword fetch bus in conjunction with the doubleword store bus. The pairing thus formed acts as a quadword bus for stride-one processing. The doubleword store bus is used for stride-*n* processing. Data are sent to the FPRs as required on the same doubleword bus. Data are sent to the GPRs as required using a separate 32-bit bus that shares internal pathing with the above.

Pipeline operation The timing of the pipelined execution of a vector instruction is variable. The performance information which follows reflects a particular model of performance, and the timings are optimistic. A paper is in preparation to document performance variations.

Each vector instruction may be thought of as composed of four sequential parts. These are start-up, element access, vector execution, and end-of-operation (end-op). Where possible, the end-op of a vector instruction is overlapped with the start-up of the following vector instruction, thereby reducing the apparent start-up time to five machine cycles. Analysis shows that start-up time can be as low as 25 cycles for many computationally intensive kernels. Element access time can be as low as zero when the vector is found in the associated data PHSB, as low as 12 cycles when the vector is found in the SHSB, and as low as 31 cycles when the vector must be accessed from central storage except when vector prefetching is in effect. Vector execution is one result per even/odd side per machine cycle.

Memory hierarchy The data PHSB associated with a Vector Facility is capable of supplying one quadword (128 bits) each machine cycle. The busing and access to data may be thought of as providing two rates, which are referred to as stride one and stride n. The stride-one rate is one quadword fetch and one queued doubleword store each machine cycle, and is achieved when consecutive data accesses to storage increase by exactly one element if the elements are in the data PHSB. There are quadword store data buffers to queue operands until such time as the data PHSB has free cycles. The stride-n rate is one element fetch and one element store each machine cycle, and is the rate achieved for all cases other than stride one.

Vector prefetching For vector stride-one and stride-two fetch operations, the machine has the ability to do special prefetch operations to reduce the cache-miss penalties. These abilities allow the fetching of two sequential data cache lines from the SHSB or CS as a single operation (a 256-byte fetch with no gaps). This can cut the number of data cache misses in half for long vectors. Additionally, any subsequent double-line fetches will be sent out to the SHSB early, overlapped with the data return from the previous double-line fetch. This reduces the apparent element access time to as few as 10 cycles, the actual inter-line gap for 256 bytes being 20 cycles. Further, these abilities reduce the comparable element access time for the case in which all data are found in the SHSB from as few as 12 cycles to as few as zero.

The above prefetching operations are used for plus-stride-one and stride-two vector fetches. Prefetching begins when the vector fetch reaches the first 256-byte boundary. Once started, the prefetching continues for the rest of the vector instruction. There is a momentary break in any vector prefetching when crossing page boundaries.

Performance The maximum theoretical performance of the ES/9000 Model 900 is 2666 MFLOPS. This is computed by multiplying the maximum number of floating-point operations per machine cycle per vector facility by the clock speed, and then multiplying by six the number of Vector Facilities in a Model 900. The maximum number of floating-point operations per machine cycle is four for either 32-bit or 64-bit execution. The rate is four for 32-bit arithmetic because either the ALU or the multiplier as required is split up to execute two 32-bit operations in parallel. The rate is four for 64-bit arithmetic when compound operations are executed that allow the ALU and the multiplier to execute simultaneously.

The maximum theoretical performance will be achieved when data in the data PHSB (or, for certain cases, data in the SHSB) are reused two or more times on the average. The memory hierarchy, as described above, delivers elements from central storage at the rate of 256 bytes in 36

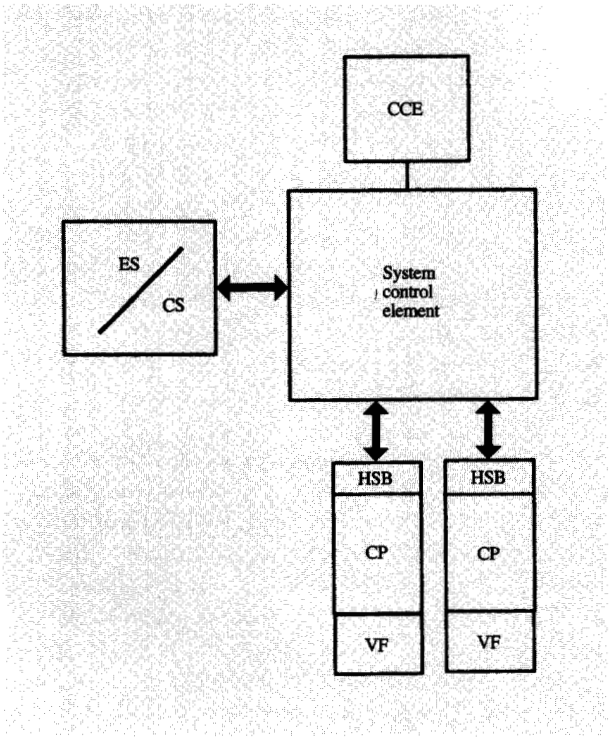

Figure 6

ES/9000 Model 480 system structure.

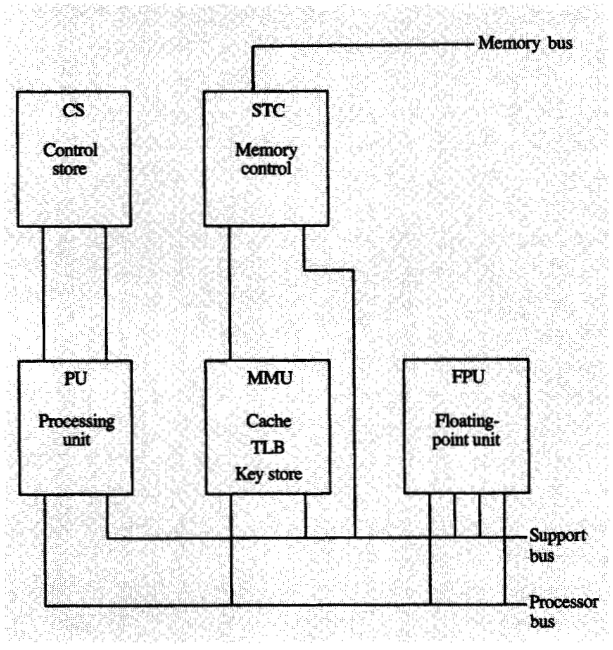

Figure 7

ES/9000 Model 170: chips and busing.

cycles when stride-one or stride-two processing occurs with no cache reuse. The maximum theoretical performance is 1185 MFLOPS for this no-cache-reuse situation.

• *ES/9000 Model 480*

An instructive example of the nine frame-mounted air-cooled ES/9000 processors [29] is the two-processor Model 480. The design of the Model 480, depicted in **Figure 6**, is based on the system structure of the IBM 3090 [30], enhanced with ESA/390 and implemented in differential current switch technology packaged in air-cooled thermal conduction modules (TCMs). The TCMs are frame-mounted in a 1.5-m^2 "footprint." The Model 480 is comparable to a 3090 Model 200 that is installable in an office environment.

The theoretical peak performance of an ES/9000 Model 480 with two Vector Facilities is 267 MFLOPS based on the cycle time of 15 ns. This system has been measured at 180 MFLOPS on the LINPACK 1000 × 1000 "best effort" benchmark; with one Vector Facility the system achieves 91 MFLOPS. The more restrictive LINPACK 100 × 100 benchmark has been measured at 22 MFLOPS using the VAST-2 preprocessor and VS FORTRAN 2.4.

Although the system structure described by Tucker [30] for the 3090 Model 200 is used for the ES/9000 Model 480, there are design differences between the two. Three of these differences are of interest for numerically intensive computing: expanded storage, cache, and floating-point multiply. The expanded storage on the Model 480 shares the memory buses with the central storage, unlike the 3090, which has separate busing for the expanded storage. Thus, the time needed for the hardware to move a page to or from Model 480 expanded storage is about 9 μs more than on the 3090. The Model 480 cache is 128 KB, twice the size of the 3090. It is doubleword-interleaved, four-way set-associative, with a 128-byte line. The floating-point multiply design of the Model 480 enables typical register-to-register multiply executions of seven cycles, as compared to four cycles on the 3090. The effect of the three differences cited is dependent upon the application, and may produce better or worse performance than the 3090. For the LINPACK 100 × 100 benchmark, these design differences are not visible in the performance result.

• *ES/9000 Model 170*

The fastest of the four original rack-mounted air-cooled ES/9000 processors is the Model 170. The system structure for a processing unit (PU) is depicted in **Figure 7**. A Model 170 has two PUs, one of which is used primarily for input/output processing. Implementation is in 40 000-circuit-per-chip CMOS technology [31] packaged in an 85-kg rack with a 0.6-m^2 footprint. The rack also has space for mounting DASD, tape, and communication I/O devices.

Four 12.7-mm^2 chips mounted on one 44-mm^2 module constitute a base ESA/390 PU. A floating-point coprocessor is implemented on two chips mounted on a second module. The base module and the coprocessor module are mounted along with supporting clock logic on one processor card. Buses between and within the chips are 8 bytes wide. The CMOS technology implementation permits a 30-ns cycle time. A 16KB four-way set-associative cache delivers 64-byte lines, and is accessible in one cycle from the working registers. Instruction processing is done in a pipeline composed of the four stages of instruction fetch, address generation, execution, and putaway.

The Model 170 has been measured at 3.7 MFLOPS on the LINPACK 100 × 100 benchmark using VS FORTRAN 2.4. It is tempting to compare a 170 with the entry-level 3090 Model 120E at 3.1 MFLOPS. Nevertheless, the ES/9000 Model 170 is advertised as an effective performance upgrade for either a 9370 Model 90 or a 4381 Model 91E, providing respectively 4.6× and 2.9× performance factors for engineers and scientists doing numerically intensive computing.

6. Potential parallel processing system

The announcement of S/390 makes possible a hardware configuration for n-way parallel processing, where n may be as large as 60. The ESCON Director™ is a nonblocking crosspoint switch with up to 60 ports.Thirty simultaneous connections, each designed to carry more than 10 MB/s between any two ports, support an aggregate data transfer rate several times that of channel speeds. This suggests the intriguing possibility of up to 60-way parallel processing.

Figure 8 depicts a parallel processing system constructed with each of 60 ES/9000 processors connecting to each of two ESCON Directors. The ES/9000 processors described in this paper are available with 12–256 ESCON channels, each with a 10MB/s bandwidth. An n-way parallel processing system for $n \leq 60$ could be configured with two channels to/from each processor. The connection would thus be channel-to-channel between any two models.

An n-way parallel processing system so constructed would operate with each attached processor running a copy of an operating system. Two obvious choices for an operating system are MVS/ESA and VM/ESA. An interesting research project [32] based on VM/ESA has been described by Amman et al. The hardware used in this project is the prototype Parallel Processing Computing System described in the IBM/CERN Joint Study Report 1990 brochure. The obvious choice for the run-time environment of such a parallel processing system would be based on the IBM Clustered FORTRAN [20] SCSE software offering.

N-way parallel processing with ES/9000 models and one or more ESCON Directors is thus a possibility that can

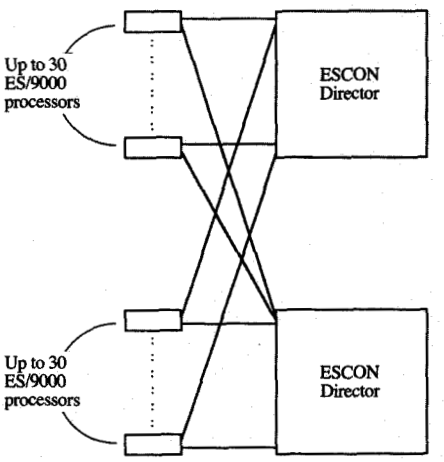

Figure 8

N-way parallel processing with ESCON Director.

and likely will be explored in the early life of S/390. The possible configurations are numerous, which surely enhances the probability of success. The supporting software that emerges will in all likelihood be based on the current state of the art. This will be a fruitful area of investigation.

7. System environments

IBM System/390 computing services are generally accessible to the engineer or scientist through a terminal. If the terminal is an intelligent workstation with an operating system, S/390 services can be invoked through cooperative processing or client/server computing. A mainframe interactive (MFI) terminal, or a workstation emulating such a terminal, invokes S/390 services for either interactive or batch computing or both.

• *Batch*

For processing on the S/390, the phrase "background processing" is more descriptive than is the older term "batch." Batch processing involves the submission of work to be executed on S/390, where the submitter specifies the data to be used, the program to be run, and certain estimates of the computing resources that will be consumed. There is usually a well-defined menu or procedure that may be used at the terminal for batch submission. Depending on the batch monitor software that the establishment chooses to run on the ES/9000 processor, the submitter may or may not be able to track

the progress of the submitted work. Background processing means that in all cases the submitter is free to use the terminal for other work once the batch submission has been accepted.

- *Interactive*

Interactive S/390 services vary depending upon the operating system chosen by the establishment where the engineer or scientist works. UNIX interactive constructs are used when AIX/ESA or AIX/370 is installed on the ES/9000 processor to which the terminal is connected. CMS and CP interactive constructs are used when the establishment selects VM/ESA. TSO constructs are available when the MVS/ESA operating system is chosen. Each of these three interactive services offers functionality to a greater or lesser degree, with unique syntax and semantics in the following categories: file naming, addressing, and manipulation; commands; enter and display. Characteristically, interactive response time varies from tenths of a second to a few seconds depending upon the service invoked by the commands or entry.

A typical terminal session for an engineer is to enter information and commands to submit batch work to the S/390, and then, while the batch work runs in the background, to use interactive services such as mail and text editing.

- *Cooperative processing, client/server computing*

The terms "cooperative processing" and "client server" computing used in the IBM System/390 literature are not self-defining. It is useful to describe six possible scenarios[3] for accessing S/390 computing services from a RISC System/6000™ workstation. The scenarios are based on software capability existing in 1991. Depending upon the intent of the reference, any one of the scenarios could be termed either cooperative processing or client/server computing.

At one extreme there is the "postprocessing" scenario. In this scenario, the ES/9000 performs all of the NIC processing, and the RISC System/6000 provides a graphical analysis of the results. At the other extreme, there is "fine-grained" cooperation, where an application runs on two or more processors with frequent interaction. Usually the intent is to reduce the elapsed time of the application.

These scenarios, and a number of intermediate scenarios, are considered below. It is assumed that if part of the application is running on (or accessing data on) the ES/9000 rather than the RISC System/6000, it is doing so for one or more of the following reasons:

- The ES/9000 can have an order of magnitude more storage.
- The absolute processing power of a multiprocessing ES/9000 system is an order of magnitude greater.
- Tape management and access are more sophisticated.
- Large databases may exist on the ES/9000.
- The user may get more service from the ES/9000 if he is one of many users on a RISC System/6000.
- The large-system environment includes access to shared resources such as DASD.

- *Postprocessing: ES/9000 to RISC System/6000 using X-Windows*

In this scenario, the NIC part of the application runs entirely on the ES/9000. The graphical part of the application is split between the ES/9000 and the RISC System/6000. The X-Client part runs on the ES/9000, and the X-Server part runs on the RISC System/6000.

The advantage of this approach is that it can be both fully interactive and postprocessing, and the user does not have to program specifically for two processors (since the client-server interface is a well-defined system-level interface).

The disadvantages are that the user is not taking advantage of the NIC processing price/performance of the RISC System/6000, that most graphical interrupts are handled by the ES/9000, and that each new image requires data to be downloaded. For a full image, one million bytes would require several seconds to be transferred over, for example, an Ethernet® connection.

- *ES/9000 to RISC System/6000: NFS*

Network File System™ (NFS™) is a protocol that uses IP (Internet Protocol) to allow cooperating computers to access one another's file systems as if they were local.

In this scenario, the main NIC application runs on the ES/9000, producing spill files that are examined by a graphics application on the RISC System/6000. The spill files are part of the ES/9000 file system. NFS is used to "mount" these files on the RISC System/6000, so that they can be accessed by the graphics application. This application uses X-Windows™ to handle all of the graphics, and can provide interactive feedback to the ES/9000 application by writing to a control file (also mounted by NFS) that is read by the main application. The NFS parameters may have to be adjusted so that the control file is physically updated on the ES/9000 as quickly as is required (within a few seconds, for example).

The advantages of this scenario are that a reasonable degree of interactivity is maintained; the graphics application does not have to deal with two processors (the application considers all files to be local); the X-Client and X-Server are both running on the RISC System/6000 (so less CPU is needed on the ES/9000, and there are fewer

[3] The first publication of these scenarios was in an IBM Internal Use publication which was the result of a residency project at the IBM International Technical Support Center, Poughkeepsie, New York. The participants were A. Barak, M. Batish, R. Bell, J. Hague, K. Wathne, and G. Wightwick.

interrupts on the ES/9000); and the data in the spill files are downloaded only once by NFS (data are kept in RISC System/6000 storage buffers if possible, and can be processed as required by the graphics application).

The main disadvantage is that two separate applications must be written; and there is still a fairly long delay in downloading the spill files (about seven seconds on Ethernet for 125 000 double-precision words).

- *ES/9000 to RISC System/6000: FTP*

File Transfer Protocol (FTP) is the Internet standard, high-level protocol for transferring files from one system to another.

In this scenario, the main NIC application runs on the ES/9000 and produces spill files that are sent by the user typing in FTP commands from the RISC System/6000. The control file is also sent back to the ES/9000 using FTP commands.

Other details, including advantages and disadvantages, are much the same as for the NFS scenario. An additional advantage is that the file is transferred only once (it may be necessary to transfer it more than once in the NFS scenario). Additional disadvantages are that the file must be written to disk, and manual synchronization of the FTP commands is required.

- *ES/9000 to RISC System/6000: Sockets*

In this scenario, the main NIC application runs on the ES/9000 and produces files that are sent directly to the RISC System/6000 using a socket connection. The control data are also sent back to the ES/9000 using socket commands.

Other details, including advantages and disadvantages, are similar to the NFS scenario above. The main additional advantages are that the data may not have to be written to disk at all, and that NFS is not involved. A main disadvantage is that more sophisticated programming, including possible error recovery, is required. (This disadvantage could be alleviated by providing well-defined subroutines.)

- *RISC System/6000: NFS*

In this scenario, the entire application runs on the RISC System/6000, but the initial data files are on the ES/9000. NFS is used to "mount" these files on the RISC System/6000 so that they can be accessed by the application.

- *Fine-grained cooperative processing*

Fine-grained cooperative processing between two or more coupled RISC System/6000s or ES/9000s or combinations thereof is taken to mean processing which requires frequent communication. The speed of the communication paths becomes important in determining whether such coupled configurations can reduce the elapsed time of an application.

If the speed of processing is very fast relative to the transfer rates over the communication paths, communication speed becomes dominant. A metric that may be used to express this is executions per byte (E/B). Numerically intensive computing is characterized by E/B ratios of 10 or more. Storage-coupled processors (for example, the ES/9000 Model 900 discussed in this paper) are characterized as having E/B capacities of 10 or less; hence, they are not limited by communication speed. Channel-coupled systems are characterized as requiring applications with E/B of 10 to 100, and network-coupled systems characteristically require an order of magnitude more procesing per byte of data transferred, needing 100 to 1000 executions per byte.

As a rule of thumb, an application must spend an order of magnitude more time processing data than transferring it.

Summary

IBM System/390 computing services of interest to engineers and scientists are based on the elements of an S/390 system reviewed and detailed in this paper. The software elements include FORTRAN, particularly the syntactical extensions for parallel computing, and UNIX. The hardware elements include ESCON and the ES/9000 processors, particularly the Model 900 design, which includes a robust memory hierarchy and the Vector Facility.

Acknowledgments

The authors wish to thank J. Wang, L. Ward, and D. Soll for their description of large applications on S/390, and S. Comfort and P. Gannon for their clear and useful description of the Model 900 memory hierarchy. The information on vector prefetching was provided by M. Ignatowski. The information on vector design was described by F. Sell, and M. Siegel, J. Stark, and G. Doettling provided the material for Model 170 design. The cooperative processing scenarios were first described by A. Barak and his co-authors. R. Clark, B. Ralston, and S. Tucker reviewed the paper and made suggestions for improvement.

National Aeronautics and Space Administration. MSC/NASTRAN is an enhanced proprietary version developed by the MacNeal/Schwendler Corporation. CATIA is a registered trademark of Dassault Systèmes, Inc. VAST-2 is a registered trademark of Pacific-Sierra Corporation. OSF/1 is a trademark of the Open Software Foundation, Inc. Ethernet is a registered trademark of Xerox, Inc. Network File System and NFS are trademarks of Sun Microsystems, Inc. X-Windows is a trademark of the Massachusetts Institute of Technology.

References and notes

1. D. H. Gibson, D. W. Rain, and H. F. Walsh, "Engineering and Scientific Processing on the IBM 3090," *IBM Syst. J.* **25**, No. 1, 36–50 (1986).
2. *C/370 Users' Guide*, Order No. SC09-1264; available through IBM branch offices. Covers input/output, interlanguage calls, and re-entrancy, among other details.
3. *APL2 Release 3 General Information Manual*, Order No. GH20-9214; available through IBM branch offices. Includes a comparison between programs written in APL and Pascal.
4. *VS PASCAL General Information Manual*, Order No. CG26-4318, and *VS PASCAL Applications Programming Guide*, Order No. SC26-4319; available through IBM branch offices.
5. *Assembler H Version 2 General Information Manual*, Order No. GC26-4035; available through IBM branch offices.
6. J. McComb and S. Schmidt, "Engineering and Scientific Subroutine Library for the IBM 3090 Vector Facility," *IBM Syst. J.* **27**, No. 4, 404–415 (1988). Also, addendum in the Technical Note: R. C. Agarwal, F. G. Gustavson, J. McComb, and S. Schmidt, "Engineering and Scientific Subroutine Library Release 3 for IBM ES/3090 Vector Multiprocessors," *IBM Syst. J.* **28**, No. 2, 345–350 (1989).
7. G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.
8. J. J. H. Forrest and J. A. Tomlin, "Vector Processing in Simplex and Interior Methods for Linear Programming," *Research Report RJ-6390*, IBM Research Division, San Jose, CA, 1988.
9. N. Karmarkar,"A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica* **4**, 373–395 (1984).
10. J. A. Tomlin, "A Note on Comparing Simplex and Interior Methods for Linear Programming," *Progress in Mathematical Programming*, N. Megiddo, Ed., Springer-Verlag, New York, 1989, pp. 91–103.
11. H. Crowder, E. L. Johnson, and M. W. Padberg, "Solving Large Scale Zero-One Linear Programming Problems," *Oper. Res.* **31**, 803–834 (1983).
12. D. Goldfarb, "A Numerically Stable Method for Solving Strictly Convex Quadratic Programs," *Math. Program.* **27**, 1–33 (1983).
13. N. D. Grigoriadis, "An Efficient Implementation of the Network Simplex Method," *Math. Program. Study* **26**, pp. 83–111 (1986).
14. *ES/9000 for the Technical Computing Environment*, Order No. G520-6784; available through IBM branch offices.
15. C. J. Georgiou, T. A. Larsen, P. W. Oakhill, and B. Salimi, "The IBM Enterprise Systems Connection (ESCON) Director: A Dynamic Switch for 200Mb/s Fiber Optic Links," *IBM J. Res. Develop.* **36**, 593–616 (1992, this issue).
16. J. J. Coleman, C. B. Meltzer, and J. L. Weiner, "Fiber Distributed Data Interface Attachment to System/390," *IBM J. Res. Develop.* **36**, 647–654 (1992, this issue).
17. *VS FORTRAN Version 2, Programming Guide, Release 5*, Order No. SC26-4222-6, and *VS FORTRAN Version 2, Language and Library Reference, Release 6*, Order No. SC26-4221-5; available through IBM branch offices.
18. C. A. Scalzi, A. G. Ganek, and R. J. Schmalz, "Enterprise Systems Architecture/370: An Architecture for Multiple Virtual Space Access and Authorization," *IBM Syst. J.* **28**, No. 1, 15–38 (1989).
19. L. J. Toomey, E. C. Plachy, R. G. Scarborough, R. J. Sahulka, J. F. Shaw, and A. W. Shannon, "IBM Parallel FORTRAN," *IBM Syst. J.* **27**, No. 4, 416–435 (1988).
20. R. J. Sahulka, E. C. Plachy, L. J. Scarborough, R. G. Scarborough, and S. W. White, "FORTRAN for Clusters of IBM ES/3090 Multiprocessors," *IBM Syst. J.* **30**, No. 3, 296–311 (1991).
21. *VAST-2 for VS FORTRAN User's Guide*, Order No. SC26-4668, Program Offering Release 1, December 1989; available through IBM branch offices. Includes a description of concepts and rules, and specific transformations.
22. G. O. Blandy and S. R. Newson, "VM/XA Storage Management," *IBM Syst. J.* **28**, No. 1, 175–191 (1989).
23. W. T. Koba and W. Iimura, *AIX/370 as a Guest Operating System*, Order No. G320-3532, December 1989; available through IBM branch offices.
24. J. M. Gdaniec and J. P. Hennessy, "VM Data Spaces and ESA/XC Facilities," *IBM Syst. J.* **30**, No. 1, 14–33 (1991).
25. *MVS/ESA Callable System Services for High-Level Languages*, Order No. GC28-1639; available through IBM branch offices.
26. *IBM Enterprise Systems Architecture/390 Principles of Operation*, Order No. SA22-7201; available through IBM branch offices.
27. *IBM Enterprise Systems Architecture/390 Vector Operations*, Order No. SA22-7125; available through IBM branch offices.
28. J. S. Liptay, "Design of the IBM Enterprise System/9000 High-End Processor," *IBM J. Res. Develop.* **36**, 713–731 (1992, this issue).
29. S. F. Hajek, "IBM Enterprise System/9000 Type 9121 Air-Cooled Processor," *IBM J. Res. Develop.* **35**, 307–312 (1991).
30. S. G. Tucker, "The IBM 3090 System: An Overview," *IBM Syst. J.* **25**, No. 1, 4–19 (1986).
31. N. Roethe, "A CMOS Implementation of the ESA/390 Mainframe Architecture," *Microprocess. & Microprogram.* **32**, 209–214 (1991).
32. E. M. Ammann, R. R. Berbec, G. Bozman, M. Faix, G. A. Goldrian, J. A. Pershing, Jr., J. Ruvolo-Chong, and F. Scholz, "The Parallel Processing Compute Server," *IBM J. Res. Develop.* **35**, No. 5/6, 653–666 (1991).

Received July 1, 1991; accepted for publication September 18, 1991

Donald H. Gibson *IBM Enterprise Systems, P.O. Box 950, Poughkeepsie, New York 12602 (retired).* Mr. Gibson recently retired as a member of the IBM Poughkeepsie Senior Technical staff in IBM Enterprise Systems. He has worked in design and development of large systems including SAGE, STRETCH, System/360™ Models 91 and 85, System 370 Models 195 and 165, the 3090, including the Vector Facility, and the ES/9000. He has lectured extensively on system design and performance of large mainframes. The "cache" design, first introduced in the IBM S/360 Model 85, evolved directly from his simulation work on block transfer memory systems design, undertaken in connection with studies of parallel systems and reported in his paper at the AFIPS Spring Joint Computer Conference in 1967. During this period Mr. Gibson lectured extensively on system design and performance of large mainframes. In the mid-1970s he pursued technical interests in graphics and artificial intelligence. The 3090 engineering/scientific design features, including both scalar and vector, are based on his definitive customer survey work in the late 1970s on large systems engineering/scientific applications. In the early 1980s, he led the activity in IBM on engineering data base, he chaired the work on a computer-aided engineering strategy for IBM, and he pioneered work on cooperative processing. His work most recently focused on various aspects of cooperative processing between intelligent workstations and large systems. Mr. Gibson received a B.S.E.E. degree from the University of Kentucky, Lexington, in 1956. He has received many IBM awards, notably the Corporate Outstanding Contribution Award in 1967 for the cache concept, and he holds several patents.

Gururaj S. Rao *IBM Enterprise Systems, P.O. Box 950, Poughkeepsie, New York, 12602 (GRAO at TDCSYS2).* Dr. Rao received his Bachelor of Engineering degree from the University of Mysore, India, his Master of Engineering degree from the Indian Institute of Science, India, and the Ph.D. degree from Stanford University, California, all in electrical engineering. He was an Assistant Professor of Electrical Engineering at Rice University, Houston, Texas, from 1975 to 1978. He joined the IBM Thomas J. Watson Research Center at Yorktown Heights, New York, in 1978, and worked on large systems processor structure studies. In 1983, Dr. Rao joined the Data Systems Division (now the Enterprise Systems Development Laboratory) in Poughkeepsie, where he is currently the manager of the Processor Architecture and System Structure Department, with responsiblity for developing future large-system requirements and architecture direction. Dr. Rao has received several academic honors as well as IBM awards. In 1991, he was appointed a Senior Technical Staff Member.