ICAP/3090 User Guide

David Folsom, Marian Klonowski, Nikos Pitsianis, Antoine Trannoy and Silvia Veronese.
With
Thierry Delbecque, Laura Evans, Pierre Ferron, Philippe Mouton and Jukka Saarinen.

IBM Corporation
Data Systems Division
Dept. 48B / MS 428
Neighborhood Road
Kingston, New York 12401
# Contents

1. Introduction ............................................ 1

2. Guide to Precompiler use .................................. 5
   2.1 Presentation ........................................... 5
   2.2 Basic Directives ....................................... 6
       2.2.1 Run-Time Configuration ............................. 8
       2.2.2 C$START (Master directive) ....................... 8
       2.2.3 C$FINISH (Master directive) ...................... 8
       2.2.4 C$EXECUTE (Master directive) ................... 8
       2.2.5 C$WAIT (Master directive) ....................... 9
       2.2.6 C$SLROUTINE (Slave directive) .................. 10
       2.2.7 C$SLxx (Slave directive) .......................... 10
       2.2.8 C$ALLOCATE (Slave directive) .................... 10
       2.2.9 C$ADDING (Master directive) .................... 12
   2.3 Shared Memory Directives ............................. 13
       2.3.1 Introduction ...................................... 13
       2.3.2 C$SHARED (Master directive) ..................... 13
       2.3.3 C$SHRxx (Master directive) ...................... 14
   2.4 Synchronization Directives. .......................... 17
       2.4.1 Introduction ...................................... 17
       2.4.2 Directives ....................................... 18
       2.4.3 Deadlock Detection ............................... 22
   2.5 Subroutines for Fine Grain Parallelism ................ 22
       2.5.1 Fetch and Add .................................... 22
       2.5.2 Add Arrays ...................................... 23

3. The ICAP/3090 User Interface .......................... 24
   3.1 Components .......................................... 24
       3.1.1 LCAP Command Processor ......................... 24
       3.1.2 Master Control Program ......................... 24
       3.1.3 Slave Control Program ........................... 24
       3.1.4 ICAP/3090 Scheduler .............................. 24
       3.1.5 Processing user's requests ..................... 26
   3.2 LCAP Command (Interactive Mode) ..................... 26
   3.3 LCAP RUN Subcommand ................................ 26
       3.3.1 RUN Entry Screen ................................ 29
       3.3.2 Controlling the Application Run ................. 30
   3.4 The Master-Slave EXEIC ............................... 30
   3.5 Aborting a Job ....................................... 31
   3.6 LCAP SIOW Subcommands ................................ 31
       3.6.1 SIOW Subcommand Menu ......................... 32
       3.6.2 SIOW SYSTEM Subcommand ....................... 32
       3.6.3 SIOW SYSS Subcommand .......................... 33
       3.6.4 SIOW JOBS Subcommand .......................... 34
       3.6.5 SIOW JOB Subcommand ............................ 34
   3.7 LCAP OPERATOR Subcommands .......................... 35
       3.7.1 SIOWALL Subcommand ............................. 35
       3.7.2 KILLJOB Subcommand ............................. 35

4. References ............................................... 36
1. Introduction

ICAP, an acronym for "loosely Coupled Array of Processors" is a Multiple-Instruction-Multiple-Data (MIMD) parallel computing system utilizing a master/slave topology. The system was originally conceived in early 1983 as a tool for a specific set of applications. However, over the course of the past 6 years the ICAP systems have evolved and developed into general purpose parallel computing systems for a broad range of scientific and engineering applications.

The first ICAP system, appropriately called ICAP-1, consisted of an IBM 4381 mainframe and 10 FPS-164 Attached Processors (APs). In this system the master was considered to be the IBM mainframe, and the APs were the slave processors. Eventually, a fast bus and shared memories between APs were added to improve communication paths between processors. A schematic representation of this system is given in Figure 1 on page 4.

In parallel to the development of the ICAP-1 system a second system was being assembled, ICAP-2. ICAP-2 was similar to ICAP-1 in the sense that it was also a master/slave system, however in the case of ICAP-2 the master was a larger IBM 3081 mainframe, and the slaves were more powerful FPS-264 Attached Processors. This machine was approximately 3-4 times more powerful than the ICAP-1 system.

Along with the development of the ICAP hardware systems was the development of the ICAP software for parallel processing. The main philosophy behind the software development was to modify the operating system software as little as possible, and to build the ICAP system on top of the IBM operating system. In this fashion the major portions of the software could be made operating system independent, and only particular pieces of the software would need modification in order to run ICAP on different operating systems.

IBM operating systems have inherent support for parallel execution on multiprocessors. In the IBM Virtual Machines VM/XA System Product operating system, each user runs his application in his own virtual machine (VM). Distinct VM's can be scheduled by the operating system to run simultaneously on different CPUs of the multiprocessor. A standard feature of the VM/XA operating system is the Inter-User Communication Vehicle (IUCV), which provides the facility to communicate messages and data between VMs. Thus, to achieve parallel execution of a single application program only requires that there be some mechanism to partition different portions of the application across several VMs, with each of the VMs connected to a master VM through the use of IUCV.

The current ICAP system, known as ICAP/3090, has been designed in an analogous fashion to the previous ICAP systems. The philosophy adopted is a master/slave topology, however in the ICAP/3090 configuration each processing node is an IBM 3090 CPU, and its associated hardware features. The central feature of ICAP/3090 is that multiple systems can be coupled to provide a two level hierarchy; "cluster" and CPU. Before we describe the configuration of the ICAP/3090 system, it is worthwhile to review the characteristics of the IBM 3090 family of processors.

The IBM 3090 vector multiprocessor family encompasses a variety of models ranging from a two-processor system (Model 200) to a six-processor complex (Model 600). Within a single 3090 it is possible to increase the throughput of a large workload by using the multiple processors on independent jobs. This has been the traditional motivation behind the development of such systems. An important corollary of this approach is that memory must be increased proportionately. This contributed to the development of expanded storage on the 3090 to alleviate the effect of paging that occurs when multiple jobs compete for real memory. Overall, the 3090 multiprocessor systems have been quite successful in increasing system throughput. It should also be pointed out that within a single IBM 3090 multi-processor all system resources such as memory, I/O, etc., can be shared by all processors. Therefore, it is also possible to decrease the turn-around time for a single job by utilizing multiple processors (and system resources) on that job. Finally, vector capabilities have been introduced to the 3090s in an integrated architectural manner with the addition of vector registers and 171 new instructions, providing each processor with scalar and vector processing. Overall, the design of this system emphasizes a balance of memory access, I/O, and processing capabilities.
The ICAP approach to parallel processing with IBM 3090s is to increase the level of parallelism available by coupling several 3090 machines together to form a system that is not massively parallel, but rather is modular and can be expanded to match the degree of parallelism that a set of applications can support. In this sense, the controlling software must also be modular and flexible, and easily adapted and expanded to take advantage of higher levels of parallelism as well as new hardware features that may become available.

The current ICAP/3090 system consists of two IBM 3090/400 Base machines, one IBM 3090/600 E machine and one IBM 3090/600 S machine, for a total of 20 processing nodes spread across four clusters. Each processing node has a scalar and vector processor, and the total memory on the four machines (main plus expanded) is 4.75 Gigabytes. Disk packs are shared by the four machines and amount to roughly 300 Gigabytes. The four 3090s are currently linked together via an IBM 3088 channel-to-channel connector which allows complete point-to-point communication among all 3090 systems. The maximum rate for transmission of data along the standard IBM channel is 4.5 Mbytes/second; therefore the maximum rate for exchange of data between IBM 3090s is of the same order of magnitude. This speed has obvious implications in determining the optimal partitioning of a parallel application, but this topic has been discussed in detail elsewhere and will not be addressed here.

The ICAP/3090 system software has been designed in a modular and flexible fashion and to support all parallel and vector capabilities on the system. Among the features worth mentioning are 31 bit addressing (2 Gbytes of virtual memory), shared memory within a single 3090 machine, parallel I/O and of course parallel processing across multiple 3090 systems.

Each parallel application uses several Virtual Machines (VM's) dedicated for the run. The VM's may be distributed across any of the 3090 systems in the ICAP complex. Each VM is initialized with a separate copy of the application program and data area, although the application may also designate special COMMON block areas in Shared Segments, which can be shared by the VM's within the same 3090 system. In this way both, distributed-memory with message passing and shared-memory facilities are available to the application.

One of the VM's is initialized first and becomes the 'Master'; controlling when parallel subroutines are executed on the remaining 'Slave' VM's. The application may have many cycles where parallel routines are initiated and completed on the Slaves. As part of each cycle, the Master controls the transfer of any necessary data to and from the Slaves. Slave VMs may each execute a different part of the 'Slave Code'. The application program has explicit control of these services through the use of ICAP Directives.

There are several parts to the ICAP system providing a sort of 'shell' around VS FORTRAN for the functions just described above:

- Precompiler and Directives
- Parallel Run Utilities
- Run Scheduler and Resource Manager
- Commands to Start, Monitor and Stop a Job
- Communications Software

Directives are special FORTRAN Comment statements which the user adds to a program to control the ICAP parallel functions. The ICAP Precompiler scans the program, converting Directives to FORTRAN statements, and producing two new source files: one file for the Master VM and a second file for the Slave VM's. Both files are compiled with the VS FORTRAN compiler.

The parallel run utilities control the initiation of parallel subroutines, along with the transfer of data between Master and Slaves.

The ICAP/3090 scheduler assigns the resources (VMs and Storage) used by an application, and initializes them for a run. A set of user commands are provided to start (or abort) a parallel run and to query on the run status. When starting a run, the user specifies how many Slaves in which system(s) the application should run on.
The following scenario will illustrate the steps used to develop an application under ICAP/3090, test it and run it for productive use.

Assume that the application exists in its sequential (non-parallel) form. The first step is deciding how to break up the program into subroutines which can be executed concurrently. The program is altered to add ICAP directives and precompiled.

When the program Precompiles without errors, the Master and Slave files are compiled by FORTRAN. The user creates the Master-Slave EXEC file to define the files and the programs loaded for the Master and Slave VM's and then the application can be executed.

The structure of this document, intended to serve as a User Guide to the ICAP/3090 system, is as follows:

Chapter two has a complete description of the Precompiler Directives and their usage with simple examples.

Chapter three describes the User Interface, the available commands, the messages-responses of the system and directions on how to write the Master-Slave EXEC with examples, together with a short description of the Scheduler-Resource Manager, its purpose and function from a user's point of view.

At the end of this document you will also find an appendix with a Quick Reference chart.