A Survey on Parallel Architecture and Parallel Programming Languages and Tools

C.Namrata Mahender
Dept. of CS and IT,
Dr.Babasaheb Ambedkar Marathwada University,
Aurangabad,India
e-mail: nam.mah@gmail.com

Abstract— In this paper, we have presented a brief review on the evolution of parallel computing to multi-core architecture. The survey briefs more than 45 languages, libraries and tools used till date to increase performance through parallel programming. We have given emphasis more on the architecture of parallel system in the survey.

Keywords- Parallel Computing, Parallelization, Classification, shared memory, distributed memory architecture

I. INTRODUCTION

In 1945 Von Neumann suggested the stored-program model of computing, the architecture stated that a program is a sequence of instructions stored sequentially in the computer’s memory and are executed one after the other in a linear, single-threaded fashion. As time went on advancement in mainframe technology expanded the idea of proposed by Von Neumann, the 1960 saw the advent of time sharing OS, run on large mainframe computer. Then the standalone PC became the buzz but due to large computation required and cost of Pc, need of concurrency came into picture, thus parallel computing became the need of time.[1,2]

Parallel computing came in the late 1950’s with advancements surfacing in the form of supercomputers throughout the 60’s and 70’s. These were shared memory multiprocessors, with multiple processors working side-by-side on shared data. In the mid 1980’s a new kind of parallel computing was launched when the Caltech Concurrent computation project built a supercomputer for scientific applications from 64 Intel 8086/8087 processors. This system showed that extreme performance could be achieved with mass market, off the shelf microprocessors. These massively parallel processor came to dominate the top end of computing, with ASCI Red supercomputer in 1997 breaking the barrier of one trillion floating point operations per seconds. Science then MPPS has continued to grow in size and power.

Staring in the late 80’s, cluster came to compete and eventually displace MPPs for many applications. A cluster is a type of parallel computer built from large numbers of off-the-shelf computers connected by an off-the-shelf network. Today, clusters are the workhorse of scientific computing and are the dominant architecture in the data centers that power the modern information age.

Today, parallel computing is becoming mainstream based on multicore processors most desktop and laptop systems now ship with multiple cores. The reason is that increasing performance through parallel processing can be far more energy-efficient than increasing microprocessor clock frequency.[2,3]

A. Era of computing

The most prominent two eras of computing are: sequential and parallel era. In the past decade parallel machines have become significant competitors to vector machines in the quest of high performance computing. A century wide view of development of computing eras is shown in figure 1. The computing era starts with development in hardware architecture, followed by system software particularly in the era of compilers and operating system, applications and reaching its saturation point with its growth in problem solving environment s. every element of computing undergoes three phases: R&D, Commercialization and commodity.[4]
II. PARALLEL COMPUTING CLASSIFICATION

Parallel computing can be defined based on architecture or the way programs are implemented, or the way the algorithm or program is decomposed.

A. Parallel computer based on Architecture

1) Flynn’s taxonomy

Flynn’s taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of Instruction and Data. Each of these dimensions can have only one of two possible states: Single or Multiple. The system has four classes SISD, SIMD, MISD, MIMD

a) Single Instruction, Single Data (SISD):
- A serial (non-parallel) computer
- Single Instruction: Only one instruction stream is being acted on by the CPU during any one clock cycle
- Single Data: Only one data stream is being used as input during any one clock cycle
- Deterministic execution
- This is the oldest and even today, the most common type of computer
- Examples: Older generation mainframes, minicomputers and workstations; most modern day PCs.

b) Single Instruction, Multiple Data (SIMD):
- A type of parallel computer
- Single Instruction: All processing units execute the same instruction at any given clock cycle
- Multiple Data: Each processing unit can operate on a different data element
- Best suited for specialized problems characterized by a high degree of regularity, such as graphics/image processing.
- Synchronous (lockstep) and deterministic execution
- Two varieties: Processor Arrays and Vector Pipelines

c) Multiple Instruction, Single Data (MISD):
- A type of parallel computer
- Multiple Instruction: Each processing unit operates on the data independently via separate instruction streams.
- Single Data: A single data stream is fed into multiple processing units.
- Few actual examples of this class of parallel computer have ever existed. One is the experimental Carnegie-Mellon C.mmp computer (1971).
- Some conceivable uses might be: multiple frequency filters operating on a single signal stream, multiple cryptography algorithms attempting to crack a single coded message.

d) Multiple Instruction, Multiple Data (MIMD):
- A type of parallel computer
- Multiple Instruction: Every processor may be executing a different instruction stream
- Multiple Data: Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Currently, the most common type of parallel computer - most modern supercomputers fall into this category.
- Examples: Most current supercomputers, networked parallel computer clusters and “grids”, multi-processor SMP computers, multi-core PCs. [4]

2) Feng’s Classification

Tse-yun Feng suggested the use of degree of parallelism to classify various computer architectures. Serial Versus Parallel Processing The maximum number of binary digits that can be processed within a unit time by a computer system is called the maximum parallelism degree P. A bit slice is a string of bits one from each of the words at the same vertical position. There are 4 types of methods under above classification

- Word Serial and Bit Serial (WSBS)
- Word Parallel and Bit Serial (WPBS)
- Word Serial and Bit Parallel (WSBP)
- Word Parallel and Bit Parallel (WPBP)

WSBS has been called bit parallel processing because one bit is processed at a time. WPBS has been called bit slice processing because m-bit slice is processed at a time. WSBP is found in most existing computers and has been called as Word Slice processing because one word of n bit processed at a time. WPBP is known as fully parallel processing in which an array on n x m bits is processed at one time.[4]

<table>
<thead>
<tr>
<th>Mode</th>
<th>Computer Model</th>
<th>Degree of Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSBS</td>
<td>The ‘MINIMA’</td>
<td>(1,1)</td>
</tr>
<tr>
<td>WPBS</td>
<td>STARAN, MPP, DAP</td>
<td>(1,256), (1,16384), (1,4096)</td>
</tr>
<tr>
<td>WSBP</td>
<td>IBM 370/168 UP, CDC6600, Burrough 7700 VAX 11/780</td>
<td>(64,1), (60,1), (48,1), (16,32,1)</td>
</tr>
<tr>
<td>WPBP</td>
<td>ILLiav IV</td>
<td>(64,64)</td>
</tr>
</tbody>
</table>

3) Handler’s Classification

Wolfgang Handler has proposed a classification scheme for identifying the parallelism degree and pipelining degree built into the hardware structure of a computer system. He considers at three subsystem levels:

- Processor Control Unit (PCU)
- Arithmetic Logic Unit (ALU)
- Bit Level Circuit (BLC)

Each PCU corresponds to one processor or one CPU. The ALU is equivalent to Processor Element (PE). The BLC corresponds to combinational logic circuitry needed to perform 1 bit operations in the ALU.
A computer system $C$ can be characterized by a triple containing six independent entities

$$T(C) = \langle K \times K', D \times D', W \times W' \rangle$$

Where $K =$ the number of processors (PCUs) within the computer,
$D =$ the number of ALUs under the control of one CPU,
$W =$ the word length of an ALU or of an PE,
$W' =$ The number of pipeline stages in all ALUs or in a PE,
$D' =$ the number of ALUs that can be pipelined,
$K' =$ the number of PCUs that can be pipelined.

### B. Parallel computer based on memory architecture

Generally system for parallel computer can be classified by using memory in two types shared memory and distributed memory

1) **Shared Memory Architecture**

Shared memory machines can be divided into two main classes based upon memory access times: UMA and NUMA.

a) **Uniform Memory Access (UMA):**

- Most commonly represented today by Symmetric Multiprocessor (SMP) machines
- Identical processors
- Equal access and access times to memory
- Sometimes called CC-UMA - Cache Coherent UMA. Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.

b) **Non-Uniform Memory Access (NUMA):**

- Often made by physically linking two or more SMPs
- One SMP can directly access memory of another SMP
- Not all processors have equal access time to all memories
- Memory access across link is slower
- If cache coherency is maintained, then may also be called CC-NUMA - Cache Coherent NUMA

2) **Distributed memory architecture**

Distributed memory systems require a communication network to connect inter-processor memory.

- Processors have their own local memory. Memory addresses in one processor do not map to another processor, so there is no concept of global address space across all processors.
- Because each processor has its own local memory, it operates independently. Changes it makes to its local memory have no effect on the memory of other processors. Hence, the concept of cache coherency does not apply.
- When a processor needs access to data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is likewise the programmer's responsibility.

- The network "fabric" used for data transfer varies widely, though it can be as simple as Ethernet.

3) **Hybrid -Distributed memory architecture**

The largest and fastest computers in the world today employ both shared and distributed memory architectures.

- The shared memory component can be a cache coherent SMP machine and/or graphics processing units (GPU).
- The distributed memory component is the networking of multiple SMP/GPU machines, which know only about their own memory - not the memory on another machine. Therefore, network communications are required to move data from one SMP/GPU to another.
- Parallel Processing based on data Parallelism
- There are many ways program is parallelized, majorly it is done in three ways instruction, task and data. In data parallel processing, full jobs are assigned for processing, it is efficient with coarse grained tasks and quasi scheduling, the figure 2 shows the different methods of data processing [3,5]

![Figure 2. Shows various Methods of parallel processing](image)

### III. PARALLEL PROGRAMMING LANGUAGE AND TOOLS

Numerous programming frameworks have been developed so far to support parallel execution in parallel computing system. The frameworks function at various levels of abstraction, represents different formal models for parallelism, exploit dedicated or general-purpose languages and vary from local area network based systems to geographically distributed systems. Each framework is specialized for a particular type of applications. Here we have given brief information about more than 45 languages and tools used to perform parallel programming.

1) **Parallel Programming**

Here we are just briefing the majorly used languages for parallel programming with for which they extension, support...
used for executing on platform and does has GUI interface.

Table 2 gives the brief about parallel languages.[6-62]

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Language</th>
<th>Language supported</th>
<th>GUI Support</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Adapter/ (Automatic Data Parallelism Translator)</td>
<td>CM, Fortran, Subset of Fortran 90</td>
<td>Athena Widget, X-Windows system</td>
<td>Language</td>
</tr>
<tr>
<td>3</td>
<td>CC++ (Compositional C++)</td>
<td>CC++</td>
<td>None</td>
<td>Language</td>
</tr>
<tr>
<td>4</td>
<td>Charm</td>
<td>C, C++</td>
<td>X Motif</td>
<td>Language + Performance Tool</td>
</tr>
<tr>
<td>5</td>
<td>Code2.0</td>
<td>Code2.0, C</td>
<td>X11R4 or X11R5</td>
<td>Language + Performance Tool</td>
</tr>
<tr>
<td>6</td>
<td>COOL (Concurrent Object Oriented Language)</td>
<td>COOL</td>
<td>None</td>
<td>Language + Performance Tool</td>
</tr>
<tr>
<td>7</td>
<td>Dino</td>
<td>C</td>
<td>None</td>
<td>Language</td>
</tr>
<tr>
<td>8</td>
<td>Fortran90</td>
<td>Fortran90</td>
<td>None</td>
<td>Language</td>
</tr>
<tr>
<td>9</td>
<td>Fortran D</td>
<td>Fortran77</td>
<td>X11R4</td>
<td>Language</td>
</tr>
<tr>
<td>10</td>
<td>Fortran M</td>
<td>FortranM</td>
<td>None</td>
<td>Language</td>
</tr>
<tr>
<td>11</td>
<td>Grads</td>
<td>Fortran77</td>
<td>X11R5</td>
<td>Language</td>
</tr>
<tr>
<td>12</td>
<td>HPF (High Performance Fortran)</td>
<td>HPF</td>
<td>None</td>
<td>Language + Performance Tool</td>
</tr>
<tr>
<td>13</td>
<td>Hypertool</td>
<td>C</td>
<td>None</td>
<td>Language + Performance Tool</td>
</tr>
<tr>
<td>14</td>
<td>Jaxe</td>
<td>C</td>
<td>None</td>
<td>Language</td>
</tr>
<tr>
<td>15</td>
<td>Linda</td>
<td>C, Fortran77</td>
<td>X11R4</td>
<td>Language + Debugger + Performance Tool</td>
</tr>
<tr>
<td>16</td>
<td>MeldC</td>
<td>MeldC</td>
<td>None</td>
<td>Languages + Debugger</td>
</tr>
<tr>
<td>17</td>
<td>Mentat</td>
<td>MPL: An Extended C++</td>
<td>None</td>
<td>Language</td>
</tr>
<tr>
<td>18</td>
<td>Modula-2*</td>
<td>Modula-2*</td>
<td>None</td>
<td>Language</td>
</tr>
<tr>
<td>19</td>
<td>OOF (Object oriented Fortran)</td>
<td>Fortran77.C++</td>
<td>None</td>
<td>Language</td>
</tr>
<tr>
<td>20</td>
<td>P-Linda (Prolog: Dlnada)</td>
<td>SR-Stas Prolog 0.7 and 2.1</td>
<td>None</td>
<td>Language</td>
</tr>
<tr>
<td>21</td>
<td>P-Languages</td>
<td>PC, P-Fortran</td>
<td>None</td>
<td>Language + Performance Tool</td>
</tr>
<tr>
<td>22</td>
<td>Parallelax</td>
<td>Parallelax</td>
<td>Macintosh</td>
<td>Language + Debugger + Performance Tool</td>
</tr>
<tr>
<td>23</td>
<td>Parallelax</td>
<td>Parallelax</td>
<td>X11R5</td>
<td>Language + Debugger + Performance Tool</td>
</tr>
</tbody>
</table>

2) Parallel Libraries and tools

The fifteen tools described in this part try to achieve portability by providing libraries. Most of them support parallelism within an application. Application-oriented high-level abstractions are provided in Canopy, whereas the others support programming languages such as C and Fortran with parallel extensions. A few support programming in both shared-memory paradigm and message-passing paradigm. Several of them extend the support for distributed memory machines to a network of computers. Table 3 shows the different libraries used till date.[62-89]

<table>
<thead>
<tr>
<th>Sr.No</th>
<th>Libraries</th>
<th>Language Supported</th>
<th>GUI Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>APPL</td>
<td>Fortran, C</td>
<td>None</td>
</tr>
<tr>
<td>2</td>
<td>Canopy</td>
<td>Fortran, C</td>
<td>None</td>
</tr>
<tr>
<td>3</td>
<td>CMI(Communication Manager)</td>
<td>C</td>
<td>None</td>
</tr>
<tr>
<td>4</td>
<td>CPS(Cooperative Processes Software)</td>
<td>Fortran, C</td>
<td>None</td>
</tr>
<tr>
<td>5</td>
<td>Express</td>
<td>Fortran 77, C</td>
<td>X-Windows, Sunview, postscript</td>
</tr>
<tr>
<td>6</td>
<td>GenMP(GENeric MultiProcessor)</td>
<td>Fortran 77</td>
<td>None</td>
</tr>
</tbody>
</table>
IV. CONCLUSION

In this paper we have discussed all major parallel architecture and its characteristics. We have discussed all major aspects of parallel computing, starting from its evolution to the present multi-core era. We covered almost all languages and libraries and tools which are used to implement parallel programs, improve performance and computing speed respectively.

REFERENCES


Application Portable Parallel Library (APPL),’’ to be published as a NASA TM, 1993.


