Architecture for large computer systems

by GEHRARD L. HOLLANDER
Hollander Associates
Fullerton, California

Evaluation criteria
As the computer field reaches maturity, users select their equipment not because of internal structure of features, but by its overall economy; i.e., performance per unit cost. Here, performance means the computer's ability to do the user's job. Cost includes not only the purchase price, but also such associated costs as space, power, operation, maintenance, software, and unreliability.1

Greater problem complexity stimulates various alternative computer-design approaches; from speeding up conventional processors to using a multiplicity of processors. But if the user is really indifferent to the internal structure provided his job is executed economically, we are confronted with an apparent contradiction. Multiple—and therefore smaller—processors violate the accepted dictum that a larger processor provides more performance per unit cost.2 According to Grosch's law,3 manufacturers price equipment to make performance proportional to the square of price; in other words, doubling the price increases the performance fourfold. However, this empirical pricing guide stemmed from an era when the manufacturer's only concern was the elasticity of the total demand for computers. In a competitive environment, manufacturing cost determines price; and it remains to be seen whether Grosch's law also applies to the cost of computers.

In a competitive environment, the user's economy becomes the guide of the successful designer. The manufacturing cost of a computer depends on the number and cost of its components. The designer must structure his components so that the fewest and least costly provide the maximum performance. The resultant computer must do the user's job with minimum time expended for overhead or housekeeping operations. Most likely, the components of the most economical computer will have the largest duty factor.4

Competitive system structures
Over the last two decades, computer economy has grown one hundred fold. The primary reason is faster circuitry, which provides more performance at the same cost when the faster circuits are in full production. A computer still faster than the largest economical unit available today could be designed by:

1. Speeding up a single instruction stream
   a. Faster single primary execution element (uniprocessor)
   b. Multiple primary execution elements (Array processor)
2. Allowing multiple instruction streams (Multiprocessor)

Uniprocessing:
Most past effort has been devoted to speeding up the primary execution element. Faster circuits and components have been cited already. Another approach is faster units, such as speeding the arithmetic element by parallel instead of serial organization and by high-speed carry. Augmenting units, such as memory look-ahead and separate I/O processors, relieve the primary execution element. Finally a larger primary memory gains time at the expense of storage. All these speed-up techniques have helped make existing conventional uniprocessors a hundred times faster during the last decade.

Array processing
A single instruction stream can be more effective if it can process several data streams simultaneously. For example, a payroll run requires essentially the same operations on the records of hundreds of employees. With an array of a hundred execution elements, the single instruction stream could cause a hundred operations to be performed at the same time.

Two distinct types of array computers have been investigated recently. In one, the associative parallel
processor, each execution element contains little more than the exclusive-or function. Because of its simplicity and low cost, each element can also serve as a storage location; and we might think of all orders being executed in memory (an associative or search memory). Such associative memory costs about one order-of-magnitude more than ordinary core memory, but several orders-of-magnitude less than a complete arithmetic unit.

An alternative array approach employs hundreds of processing elements as complex as a small computer. These elements can execute more complex instructions when stimulated by the primary instruction stream. The processing element’s own stored program can be tailored to its specific data stream.

Either form of array processing appears best when the problem itself has geometric properties compatible with the topology of the processing array. When the problem does not have this topological property naturally—for example, inventory or payroll problems—the problem can often be converted to use multiple execution elements effectively.

Array-type processing is inefficient if most processing elements are idle most of the time. In the associative processor, with essentially no decision capability at each processing element, the operations must be identical to the simplest level. The processor arrays, with significant decision-making and logical capability at each execution element, can allow reasonable differences from one execution element to another. However, if most decisions make an element idle, the efficiency will be quite low.

**Multiprocessing**

The multi-processor computer contains several computers, each capable of executing the job alone. They attack a complex problem by segmenting it into many parts and executing each segment autonomously. This allows dynamic allocation of the program among the available processors and provides protection against system failure if one of the processors fails.

Multiprocessor-computer task assignments require overhead, either as added hardware or as additional (unproductive) time. Furthermore, if Groch’s law holds also for computer cost, substituting several small processors for a large one starts the trade off in a less favorable performance-per-unit-cost regime.

**Comparison of structures**

Table I contrasts key characteristics of the four structures. All but the uniprocessor employ multiple data streams, either as multiple instruction streams with a single data stream, or as single data streams of multiple instruction streams.

Multiple data streams controlled by a single instruction stream must be similar. This restriction is quite severe for the associative processor, because the instruction stream is at the level of Boolean functions. The limitations are strong even for the processor array, where the main instruction stream is closer to a macro-order. While the macro level allows the stored program to adjust each execution element for some data-stream differences, the individual elements must execute the same macro operations or remain idle. For the multiprocessor, the running time of the segments should be within one or two orders-of-magnitude, a minor restriction.

The cost-division values between the portions of the system represent an approximate indication of the processing components for hypothetical “typical” systems. In the uniprocessor, the entire cost is in the central processor. The multiprocessor system has no

---

Table I. System-Structure Characteristics

<table>
<thead>
<tr>
<th>Structure*</th>
<th>Instruction streams</th>
<th>Data streams per Instruction stream</th>
<th>Limitation on data-stream similarity</th>
<th>Approximate cost division</th>
<th>Level of work division</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniprocessor</td>
<td>Single</td>
<td>Single</td>
<td>None</td>
<td>Central processor 0.0</td>
<td>Total 1.0</td>
</tr>
<tr>
<td>Multi-Processor</td>
<td>Multiple</td>
<td>Single</td>
<td>Minor</td>
<td>Inter-connection 0.3</td>
<td>Total 0.7</td>
</tr>
<tr>
<td>Associative Processor</td>
<td>Single</td>
<td>Multiple</td>
<td>Severe</td>
<td>Other execution elements 0.4</td>
<td>Total 0.7</td>
</tr>
<tr>
<td>Processor Array</td>
<td>Single</td>
<td>Multiple</td>
<td>Strong</td>
<td>0.2</td>
<td>Each 0.1</td>
</tr>
</tbody>
</table>

*Table I treats pure structures. Some structures in the references are hybrids.
central processing component, as each processor is one of the execution elements. However, a significant portion of the cost is in the interconnection and the task assignment. These overhead functions, hardware or execution time, represent approximately 30 per cent.

For the associative processor, the associative memory usually represents less than half of the system cost. For the processor array of the SOLOMON type, most of the cost is in the execution elements. The cost portion of each processing element illustrates the wide range between uniprocessors where the single processing element represents essentially the entire cost, to the associative processors where an element represents on the order of $10^5$ to $10^4$ of the cost.

Efficient use of any unconventional processor calls for new programming approaches. Since during the last two decades people have been trained to think serial, at least now the programming for the less conventional systems is more difficult until now procedures are developed. Thirty years ago, would it have been more difficult to train people to think parallel instead of serial?

**Which structure is best?**

No one structure is best for all possible jobs. The controversy revolves around the relative advantages of each architecture for different applications.

Each of the companion papers\(^4,5,6,8\) attempts to show the superiority of its approach for some or many applications. In fact, Amdahl believes the uniprocessor is best for all but a few special-purpose applications. Table II summarizes each protagonist’s view of the appropriate application areas for the different computer structures. Their papers state their reasoning. Unfortunately, or fortunately, an “unbiased” chairman cannot inject his own conclusions into a debate. The four authors speak well for themselves.

**Rating Key:**

1. Best
2. Acceptable, depending on the application
3. Marginal, depending on the application
4. No reasonable man would consider

Each companion-paper author rated the merit of each structure for the different applications, each large enough to require approximately $10^7$ executions per second with a conventional processor. The *rating* columns show the average of their ratings. Unless the value is either near 1 or near 4, it probably does not represent a consensus but an average of divergent opinions. The other column in each group shows how many respondents considered this the best structure for the application class.

**REFERENCES**

1. **G L Hollander**
   *Drum Organization for Strobe Addressing*
   IRE Transactions on Electronic Computers Vol EC 10 No 4 December 1961 p 722

2. **K E Knight**
   *Changes in Computer Performance*
   Datamation Vol 12 No 19 September 1966 pp 40-54

3. **H R Grosch**
   *High Speed Arithmetic*
   The Digital Computer as a Research Tool Journal of the Optical Society of America Vol 43

4. **R H Fuller**
   *Associative Parallel Processing*
   AFIPS Conf Proc Vol 30 April 1967

---

**Table II. Consensus of Protagonists**

<table>
<thead>
<tr>
<th>Application</th>
<th>Uniprocessor</th>
<th>Multi-processing</th>
<th>Associative Processor</th>
<th>Processor array</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ave. rating</td>
<td>No. of bests</td>
<td>Ave. rating</td>
<td>No. of bests</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ave. rating</td>
<td>No. of bests</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ave. rating</td>
<td>No. of bests</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ave. rating</td>
<td>No. of bests</td>
</tr>
</tbody>
</table>

**From the collection of the Computer History Museum (www.computerhistory.org)**
5 D L SLOTNICK
*Unconventional Systems*
AFIPS Conf Proc Vol 30 April 1967

6 G P WEST
*The Best Approach to a Large Computing Capability*
AFIPS Conf Proc Vol 30 April 1967

7 G L HOLLANDER
*Multiprocessor Systems Organization*
IEEE International Convention Record

8 G M AMDALH
*Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities*
AFIPS Conf Proc Vol 30 April 1967

9 F GRUENBERGER
*Are Small Free standing Computers Here to Stay*
Datamation Vol 12 No 4 April 1966 pp 67-68