**Von Neumann Bottleneck**

The von Neumann bottleneck looks at how to serve a faster CPU by allowing faster memory access. Part of the basis for the von Neumann bottleneck is the von Neumann architecture, in which a computer stores programming instructions, along with actual data, versus a Harvard architecture, where these two kinds of memory are stored separately. These types of setups became necessary as simpler, preprogrammed machines gave way to newer computers requiring better ways to control programming and information data.

Computer scientists have attempted to address the von Neumann bottleneck in various ways. One is to place critical memory in an easily accessible cache. There is also the idea of multithreading, or managing multiple processes in a triaged system. Other potential tools, like parallel processing, or changing the memory bus design, also work on the idea of decreasing this "bottleneck" or, in a phrase commonly used with this issue, increase the bandwidth for memory coming in and out of the processor.

Other ideas for "fixing" a von Neumann bottleneck are more conceptual. Experts have posited various "non-von Neumann" or "non-von" systems, some modeled around the biological world, which would allow for more distributed memory intake, versus the linear system used in conventional computing. Some ideas involve other emerging technologies, such as where a "memrister" or other nanoscale component could help with memory processing. The diversity of ideas around the von Neumann bottleneck show how integral this idea is to evaluating computing's potential as it has emerged over the last few decades.

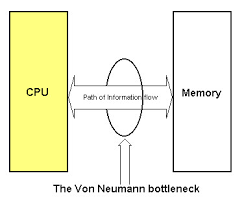


Fig 1 : von Neumann machine – The connection between processor and memory is

a bottleneck

There are a number of potential workarounds for von Neumann’s Bottleneck, including

**-Caching**: Allows data to be more accessible than if it were stored in main memory

**-Prefetching**: Making certain data available via cache before requested to make it more accessible if a request is submitted

-**Parallel Processing**: Using more than one flow of instructions to complete a computation

-**Multithreading**: Using separate threads to manage several requests at the same time

- **Pipelining**: A set of data processing elements is connected in a series, so the output of one element is the input of the next one; elements may be executed in parallel

-**Processing in Memory**: Integrating a processor and memory into a single microchip

-**Changing the Memory Bus Design**: This involves increasing the bandwidth for memory coming in and out of the processor

**Conclusion**:

Although bottlenecks in production systems have been studied for more than ten years, future directions are still promising. A common definition is possible by parameterizing the demands, and the detection methods may provide more accurate and reliable detection with the advantage of the data mining technology and computing powers. Bottleneck based analysis will be more important in understanding complex system’s behavior, and will probably be extended to the internet traffic scheduling and wireless channel allocation.

Ref 1 : <http://hartenstein.de/cited/Damian_Millers-Award.pdf>

Ref 2: <https://www8.cs.umu.se/kurser/5DV011/VT10/slides/pds02.pdf>

Ref 3 :<https://www.npmjs.com/package/bottleneck>