Oraye Godspower *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 7, July- 2022, pg. 16-26 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

EVOLUTION OF SUPERCOMPUTER ARCHITECTURE: A SURVEY

Oraye Godspower[1]; V.I.E Anireh[2]

[1,2]Computer Science Department, Rivers State University, Port Harcourt, Nigeria

oraye.godspower@ust.edu.ng

anireh.ike@ust.edu.ng

DOI: 10.5281/zenodo.6981367

ABSTRACT

Approaches to supercomputer architecture have taken dramatic turns since the earliest systems were introduced in the 1960s. Early supercomputer architectures pioneered by Seymour Cray relied on compact innovative designs and local parallelism to achieve superior computational peak performance. However, in time the demand for increased computational power ushered in the age of massively parallel systems. While the supercomputers of the 1970s used only a few processors, in the 1990s, machines with thousands of processors began to appear and by the end of the 20th century, massively parallel supercomputers with tens of thousands of "off-the-shelf" processors were the norm. Supercomputers of the 21st century can use over 100,000 processors (some being graphic units) connected by fast connections. Throughout the decades, the management of heat density has remained a key issue for most centralized supercomputers. The large amount of heat generated by a system may also have other effects, such as reducing the lifetime of other system components. There have been diverse approaches to heat management, from pumping Fluorinert through the system, to a hybrid liquid-air cooling system or air cooling with normal air conditioning temperatures. Systems with a massive number of processors generally take one of two paths: in one approach, e.g., in grid computing the processing power of a large number of computers in distributed, diverse administrative domains, is opportunistically used whenever a computer is available. In another approach, a large number of processors are used in close proximity to each other, e.g., in a computer cluster. In such a centralized massively parallel system the speed and flexibility of the interconnect becomes very important, and modern supercomputers have used various approaches ranging from enhanced Infiniband systems to three-dimensional torus interconnects.

KEYWORDS: parallelism, architecture, processing, supercomputers, interconnect

1. Introduction

Early Systems with a Few Processors (Mid 1970s and 1980s)

The Control Data Cooperation (CDC) 6600 series of computers were very early attempts at supercomputing and gained their advantage over the existing systems by relegating work to peripheral devices, freeing the Central Processing Unit (CPU) to process actual data. With the Minnesota FORTRAN compiler the 6600 could sustain 500 kiloflops on standard mathematical operations. (Frisch and Michael, 1972)

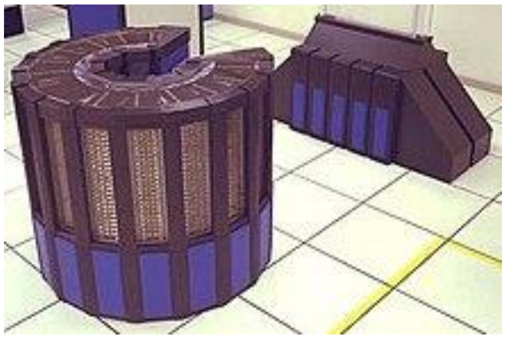

Figure 1. Cylindrical shape of the early Cray computers

The cylindrical shape of the early Cray computers centralized access, keeping distances short and uniform. (Hill *et al.*, 2000)

Other early supercomputers like the Cray 1 and Cray 2 that showed up a while later utilized few quick processors that worked as one and were consistently associated with the biggest measure of shared memory that could be overseen at the time. (Hill et al., 2000)

These early models presented equal handling at the processor level, with developments, for example, vector handling, in which the processor can play out a few tasks during one clock cycle, as opposed to sitting tight for progressive cycles.

In time, as the quantity of processors expanded, different engineering issues arose. Two issues that should be tended to as the quantity of processors increments are the dissemination of memory and handling. In the disseminated memory approach, every processor is truly bundled close with some nearby memory. The memory related with different processors is then "further away" in view of transfer speed and idleness boundaries in non-uniform memory access.

During the 1960s pipelining was seen as an advancement, and by the 1970s the utilization of vector processors had been deeply grounded. By the 1980s, numerous supercomputers utilized equal vector processors. (Hoffman and Allan 1989)

The relatively small number of processors in early systems, allowed them to easily use a shared memory architecture, which allows processors to access a common pool of memory. In the early days a common approach was the use of uniform memory access (UMA), in which access time to a memory location was similar between processors. The use of non-uniform memory access (NUMA) allowed a processor to access its own local memory faster than other memory locations, while cache-only memory architectures (COMA) allowed for the local memory of each processor to be used as cache, thus requiring coordination as memory values changed.

As the quantity of processors increments, effective interprocessor communication and synchronization on a supercomputer turns into a test. Various methodologies might be utilized to accomplish this objective. For example, in the mid-1980s, in the Cray X-MP framework, shared registers were utilized. In this methodology, all processors approached shared registers that didn't move information to and fro however were just utilized for interprocessor correspondence and synchronization. Nonetheless, inborn difficulties in dealing with a lot of divided memory between numerous processors brought about a transition to additional conveyed models. (Dongarra *et al.*,1995)

2. Massive Centralized Parallelism (1990s)

During the 1980s, as the demand for computing power increased, the trend to a much larger number of processors began, ushering in the age of massively parallel systems, with distributed memory and distributed file systems, given that shared memory architectures could not scale to a large number of processors.(Hoffman and Allan, 1989) Hybrid approaches such as distributed shared memory also appeared after the early systems.(Jelica *et al.*,1998)

Figure 2. Massive Centralized Parallelism

The computer clustering approach connects a number of readily available computing nodes (e.g. personal computers used as servers) via a fast, private local area network. The activities of the computing nodes are orchestrated by "clustering middleware", a software layer that sits atop the nodes and allows the users to treat the cluster as by and large one cohesive computing unit, e.g. via a single system image concept. (Tomoya *et al.*, 2007)

PC clustering depends on a centralized management approach which makes the nodes accessible as organized shared servers. It is unmistakable from different methodologies, for example, shared or matrix registering which likewise utilizes numerous hubs, yet with an undeniably more disseminated nature. By the 21st 100 years, the TOP500 association's semiannual rundown of the 500 quickest supercomputers frequently incorporates many groups,

At the point when countless nearby semi-free figuring hubs are utilized (for example in a bunch design) the speed and adaptability of the interconnect turns out to be vital. Present day supercomputers have adopted various strategies to resolve this issue, for example Tianhe-1 purposes an exclusive rapid organization in view of the InfiniBand QDR, improved with FeiTeng-1000 CPUs. Then again, the Blue Gene/L framework utilizes a three-layered torus interconnect with helper networks for worldwide interchanges. In this approach every hub is associated with its six closest neighbors. A comparative torus was utilized by the Cray T3E. (Adiga et al., 2005)

Massive centralized systems at times use special-purpose processors designed for a specific application, and may use field-programmable gate arrays (FPGA) chips to gain performance by sacrificing generality.

Examples of special-purpose supercomputers include Belle, Deep Blue, and Hydra, for playing chess, Gravity Pipe for astrophysics,(Makino and Makoto, 1998) MDGRAPE-3 for protein structure computation molecular dynamics and Deep Crack, for breaking the DES cipher

3. Massive Distributed Parallelism

Grid computing uses a large number of computers in distributed, diverse administrative domains. It is an opportunistic approach which uses resources whenever they are available.

(Prodan and Thomas, 2007) An example is BOINC a volunteer-based, opportunistic grid system. Some BOINC applications have reached multi-petaflop levels by using close to half a million computers connected on the internet, whenever volunteer resources become available. However, these types of results often do not appear in the TOP500 ratings because they do not run the general purpose Linpack benchmark.

Although grid computing has had success in parallel task execution, demanding supercomputer applications such as weather simulations or computational fluid dynamics have remained out of reach, partly due to the barriers in reliable sub-assignment of a large number of tasks as well as the reliable availability of resources at a given time.

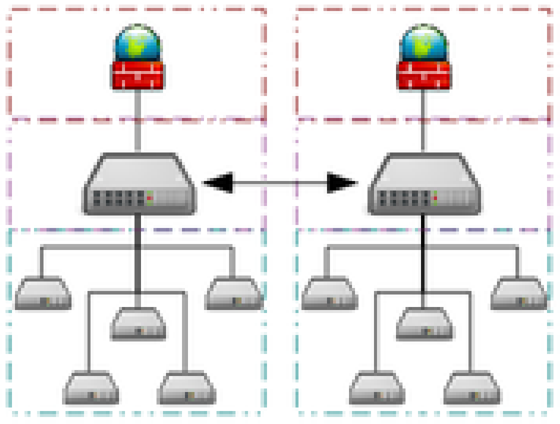

Figure 3. Massive Distributed Parallelism

In quasi-opportunistic supercomputing a large number of geographically disperse computers are orchestrated with built-in safeguards. The quasi-opportunistic approach goes beyond volunteer computing on a highly distributed systems such as BOINC, or general grid computing on a system such as Globus by allowing the middleware to provide almost seamless access to many computing clusters so that existing programs in languages such as Fortran or C can be distributed among multiple computing resources.(Kravtsov *et al.*, 2007)

Quasi-opportunistic supercomputing aims to provide a higher quality of service than sharing. The quasi-opportunistic approach enables the execution of demanding applications within computer grids by establishing grid-wise resource allocation agreements; and fault tolerant message passing to abstractly shield against the failures of the underlying resources, thus maintaining some opportunism, while allowing a higher level of control.

4. The 21st-Century Architectural Trends

The air-cooled IBM Blue Gene supercomputer architecture trades processor speed for low power consumption so that a larger number of processors can be used at room temperature, by using normal air-conditioning. The second-generation Blue Gene/P system has processors with integrated node-to-node communication logic. It is energy-efficient, achieving 371 MFLOPS/W.

Figure 4. 21st century Architecture

The K computer is a water-cooled, homogeneous processor, distributed memory system with a cluster architecture. It uses more than 80,000 SPARC64 VIIIfx processors, each with eight cores, for a total of over 700,000 cores—almost twice as many as any other system. It comprises more than 800 cabinets, each with 96 computing nodes (each with 16 GB of memory), and 6 I/O nodes. Although it is more powerful than the next five systems on the TOP500 list combined, at 824.56 MFLOPS/W it has the lowest power to performance ratio of any current major supercomputer system. The follow up system for the K computer, called the PRIMEHPC FX10 uses the same six-dimensional torus interconnect, but still only one processor per node.

Unlike the K computer, the Tianhe-1A system uses a hybrid architecture and integrates CPUs and GPUs. It uses more than 14,000 Xeon general-purpose processors and more than 7,000 Nvidia Tesla general-purpose graphics processing units (GPGPUs) on about 3,500 blades. It has 112 computer cabinets and 262 terabytes of distributed memory; 2 petabytes of disk storage are implemented via Lustre clustered files. Tianhe-1 uses a proprietary high-speed communication network to connect the processors. The proprietary interconnect network was based on the Infiniband QDR, enhanced with Chinese made FeiTeng-1000 CPUs. In the case of the interconnect the system is twice as fast as the Infiniband, but slower than some interconnects on other supercomputers.

The limits of specific approaches continue to be tested, as boundaries are reached through large scale experiments, e.g., in 2011 IBM ended its participation in the Blue Waters petaflops project at the University of Illinois. The Blue Waters architecture was based on the IBM POWER7 processor and intended to have 200,000 cores with a petabyte of "globally addressable memory" and 10 petabytes of disk space. (Biswas and Rupak, 2010) The goal of a sustained petaflop led to design choices that optimized single-core performance, and hence a lower number of cores. The lower number of cores was then expected to help performance on programs that did not scale well to a large number of processors. (Biswas and Rupak, 2010) The large globally addressable memory architecture aimed to solve memory address problems in an efficient manner, for the same type of programs. Blue Waters had been expected to run at sustained speeds of at least one petaflop, and relied on the specific water-cooling approach to manage heat. In the first four years of operation, the National Science Foundation spent about $200 million on the project. IBM released the Power 775 computing node derived from that project's technology soon thereafter, but effectively abandoned the Blue Waters approach.

Architectural tests are going on in various headings, for example the Cyclops64 framework utilizes a "supercomputer on a chip" approach, toward a path away from the utilization of enormous distributed processors. Each 64-digit Cyclops64 chip contains 80 processors, and the whole framework utilizes a worldwide addressable memory design. (Hai and Daniel, 2005) The processors are associated with non-inside obstructing crossbar switch and communicate with one another through worldwide interleaved memory. There is no data cache in the architecture; however 50% of each SRAM bank can be utilized as a scratchpad memory. (Hai and Daniel, 2005) Although this kind of design permits unstructured parallelism in a powerfully non-bordering memory framework, it likewise delivers difficulties in the proficient planning of equal calculations to a many-center framework. (Tan et al., 2009)

REFERENCES

[1] Frisch., Michael, J., (1972). Remarks on algorithm 352 [S22], algorithm 385 [S13], algorithm 392 *Communications of the ACM*. **15** (12): 1074.
[2] Hill, Mark D., Jouppi, Norman P., Sohi, Gurindar. (2000). Readings in computer architecture. pp. 40–49.
[3] Hoffman, Allan R. (1989). Supercomputers: directions in technology and applications. Washington, D.C.: National Academy Press. pp. 35–47.
[4] El-Rewini, Hesham, Mostafa Abd-El-Barr (2005). Advanced computer architecture and parallel processing. Hoboken,NJ: Wiley-Interscience. pp. 77–80.
[5] Dongarra,J., Grandinetti, L., Kowalik, L., Joubert, G,R., (1995). High Performance Computing: Technology, Methods and Applications.
[6] Jelica Protić., Milo Tomasevic., Veljko Milutinović. (1998). Distributed shared memory: concepts and systems. *IEEE Computer Society Press*. pp. 6–16.
[7] Tomoya Enokido., Leonard Barolli., Makoto Takizawa. (2007). Network-based information systems: first international conference, NBiS 2007, Regensburg, Germany, September 3-7, 2007: proceedings. Berlin: Springer. p. 375.
[8] Adiga, N. R.., Blumrich, M. A., Chen, D., Coteus, P., Gara, A., Giampapa, M. E., Heidelberger, P., Singh, S., Steinmacher-Burow, B. D., Takken, T., Tsao, M., Vranas, P., (2005). Blue Gene/Ltorus interconnection network" *Journal of Research and Development*. **49** (2.3): 265–276
[9] Makino, Junichiro, Makoto Taiji (1998). Scientific simulations with special purpose computers: the GRAPE systems. Chichester
[10] Prodan, Radu, Thomas Fahringer (2007). Grid computing experiment management, tool integration, and scientific workflows. Berlin: Springer. pp. 1–4.
[11] Kravtsov, Valentin, David Carmeli, Werner Dubitzky, Ariel Orda, Assaf Schuster, Benny Yoshpa (2007). "Quasi- opportunistic supercomputing in grids". *IEEE* International Symposium on High Performance Distributed Computing: 233–244.
[12] Tan, Guangming, Sreedhar, Vugranam C., Gao, Guang R. (2009). "Analysis and performance results of computing betweenness centrality on IBM Cyclops64". *The Journal of Supercomputing*. **56** (1): 1–24.
[13] Hai Jin., Daniel A., Reed, Wenbin Jiang., (2005). Network and Parallel Computing: IFIP International Conference, NPC 2005, Beijing, China, Birkhäuser. 132–133.
[14] Biswas, Rupak (2010). Parallel computational fluid dynamics: recent advances and future directions: papers from the 21st International Conference on Parallel Computational Fluid Dynamics. Lancaster, *DEStech Publications*. p.401.