EXASCALE COMPUTING:

ADVANCING PARALLELISM IN COMPUTING

TABLE OF CONTENTS

CHAPTER 1

ABSTRACT

Embarking on the exascale computing journey signifies more than a mere increase in computational power; it's a pivotal evolution in computing architecture. This seminar explores the imminent shift towards exascale, characterized by a staggering increase in parallelism involving millions of processors.

The ripple effects extend beyond raw processing power, prompting radical changes in hardware design, driven not only by technological aspirations but also by the pragmatic constraints of power consumption. Delving into the heart of this transformation, to examine the intricate interplay between hardware and software—from the intricacies of application codes to the orchestration of compilers, I/O, middleware, and related software tools.

As we navigate this uncharted territory, it becomes crucial to draw insights from past transitions, such as the journey from the megaflop to the present petaflop era. This retrospective lens allows us to glean valuable lessons, providing a foundation for understanding the challenges and opportunities on the road to exascale. Furthermore, we explore the readiness of advanced applications to fully harness the transformative potential of exascale computing.

As we push the boundaries of computational capability, the quest for exascale computing also underscores the critical importance of addressing ethical and societal implications. The immense power of exascale computing raises questions about data privacy, security, and the responsible use of AI and machine learning. This seminar will delve into these complex issues, aiming to foster a holistic understanding of exascale computing's impact on society and the ethical considerations that must accompany such technological advancements.

CHAPTER 2

INTRODUCTION

2.1 GENERAL

Exascale computing stands at the forefront of high-performance computing, representing a monumental leap in computational power. It refers to systems capable of performing a quintillion (10^18) calculations per second, known as an exaflop. Achieving exascale computing entails harnessing the power of millions of processing cores working in concert.

A FLOP (Floating Point Operation) is a basic arithmetic operation involving floating point numbers, typically addition, subtraction, multiplication, or division. In the context of computing performance, FLOPs are used as a measure of computational speed, indicating the number of floating-point operations a processor can perform in a given amount of time.

To put this into perspective, consider that a standard laptop operates at the level of gigaflops (10^9 floating-point operations per second), and current supercomputers typically operate in the petaflop (10^15) range. Exascale computing represents a thousand-fold increase in performance over current supercomputers, enabling researchers to tackle complex problems at an unprecedented scale and speed.

2.1.1 FACTORS DRIVING EXASCALE COMPUTING

•**Increased Core Counts:** Enhanced parallelism through higher core counts enables moresimultaneous computations, crucial for exascale performance.

•**ARM-Based Architectures:** ARM processors offer energy-efficient performance, suitable forscaling to exascale levels.

•**System-on-Chip (SoC) Designs:** Integration of multiple components into a single chip reduceslatency and power consumption, vital for exascale systems

•**Advanced Memory Technologies:** High Bandwidth Memory (HBM), Non-Volatile MemoryExpress (NVMe), and 3D-stacked memory enhance data access speeds and efficiency.

•**High-Bandwidth Interconnects:** Technologies like Infinity Fabric provide fast data transferbetween components, essential for exascale systems.

•**GPU Computing:** GPU architectures, such as CUDA and Tensor Cores, offer massive parallelprocessing capabilities, aiding exascale performance

.•Energy Efficiency: Focus on compute power per watt ensures sustainable and cost-effectiveexascale systems

.•Heat Management: Techniques like vapor chambers, phase change cooling, liquid cooling,graphite sheets, and liquid metals manage heat dissipation efficiently, crucial for exascalesystems.

•**Low Latency:** Minimizing latency in data transfer and computation is essential for real-timeprocessing in exascale systems

.•Advanced Processor Designs: Technologies like AMD's HEDT and Intel's performance hybriddesigns optimize performance and energy efficiency for exascale computing.

•**Shrinking Process Nodes:** Smaller process nodes offer improved performance and efficiency,benefiting exascale systems.

•**RISC Architecture:** RISC (Reduced Instruction Set Computer) designs, as seen in Apple Mseries and Qualcomm Snapdragon Elite X, optimize performance for specific tasks, aidingexascale computing.

•**Clock Speed Optimization:** Increasing base and boost clock speeds, including overclocking,boosts computational performance, useful for exascale computing.

•**Software Optimization:** Tools like Intel Turbo Boost and MSI Afterburner optimize CPUparameters and GPU performance, enhancing exascale computing capabilities.

•**Unified Memory:** Unified memory architectures and high-frequency DDR memory enhancedata access speeds and efficiency in exascale systems.

2.1.2 EXASCALE COMPUTE AND BIG DATA -

Exascale computing represents a pivotal advancement in computing power, enabling unparalleled speed and efficiency in processing vast amounts of data. This capability revolutionizes big data analytics by offering the scalability needed to handle massive data sets, previously deemed too large for comprehensive analysis. With exascale computing, organizations can extract insights from data in real time, facilitating quicker and more informed decision-making. This technology also facilitates complex data analysis, empowering advanced machine learning algorithms and deep learning models to uncover intricate patterns and insights from data, leading to more accurate predictions and strategic decisions. However, along with these advantages, exascale computing also presents challenges, particularly in terms of data security and privacy. Organizations must implement robust security measures to protect sensitive data during processing and analysis. Additionally, exascale computing requires significant infrastructure investments, including high-performance computing systems and storage solutions. Despite these challenges, the combination of exascale computing and big data analytics holds immense potential for driving innovation and discovery across various industries, offering new insights, product innovations, and business opportunities.

Furthermore, the integration of exascale computing and big data analytics is reshaping traditional approaches to data management and analysis. This convergence allows for the development of more sophisticated algorithms and models that can uncover deeper insights from complex datasets. Machine learning and artificial intelligence algorithms, powered by exascale computing, can process and analyze data in ways that were previously impossible, leading to more accurate predictions and insights. Additionally, the ability to analyze data in real-time enables organizations to respond quickly to changing market conditions, identify emerging trends, and make informed decisions. As this integration continues to evolve, it is expected to drive significant advancements in various fields, including healthcare, finance, manufacturing, and more.

2.2 LITERATURE SURVEY

Title – Massive Parallel Computational Model for Heterogeneous Exascale Computing System

Author – Muhammad Usman Ashraf, Fathy Alboraei Eassa, Aiiad Ahmad Albeshri

Year – 2017

Description – In next half decade, a drastic change is anticipated in High Performance Computing (HPC) architectures to achieve emerging Exascale computing system which will be thousand-fold increase in current Petascale computing. This heterogeneous (CPUs + GPUs) framework based exascale computing system will be capable to perform one ExaFlops (10^18) calculation per second. The pioneers in HPC have defined some limitations including energy (<;20 MW), budget (<;100 million $) and time (2020) that leads the number of challenges for targeted technology system. Nevertheless, an early initiative is important that have been taken by different development communities and vendors. Looking forward the tremendous challenges on the road of Exascale, massive parallelism is one of them which requires a new Parallel Programming (PP) model to provide the performance level required for exascale computing. In this paper we have proposed a new hybrid PP model that provides the coarse-grain and fine-grain parallelism at both intra-node and inter-node levels through heterogeneous architecture. The proposed model could be considered a leading model to fulfill performance demand for exascale computing system.

Title – Scaling Support Vector Machines Towards Exascale Computing for Classification of Large-Scale High-Resolution Remote Sensing Images

Author – Ernir Erlingsson, Gabriele Cavallaro, Morris Riedel, Helmut Neukirchen

Year – 2018

Description – Progress in sensor technology leads to an ever-increasing amount of remote sensing data which needs to be classified in order to extract information. This big amount of data requires parallel processing by running parallel implementations of classification algorithms, such as Support Vector Machines (SVMs), on High-Performance Computing (HPC) clusters. Tomorrow's supercomputers will be able to provide exascale computing performance by using specialized hardware accelerators. However, existing software processing chains need to be adapted to make use of the best fitting accelerators. To address this problem, a mapping of an SVM remote sensing classification chain to the Dynamical Exascale Entry Platform (DEEP), a European pre-exascale platform, is presented. It will allow to scale SVM-based classifications on tomorrow's hardware towards exascale performance.

Title – Towards Exascale Computing for Autonomous Driving

Author – Levent Gürel

Year – 2018

Description - Autonomous driving is emerging as one of the most computationally demanding technologies of modern times. We report a high-level roadmap towards satisfying the ever-increasing computational needs and requirements of autonomous mobility that is easily at exascale level. We demonstrate that hardware solutions alone will not be sufficient, and exascale computing demands should be met with a combination of hardware and software. In that context, algorithms with reduced computational complexities will be crucial to provide software solutions that will be integrated with available hardware, which is also limited by various mobility restrictions, such as power, space, and weight.

Title – Portable and Scalable All-Electron Quantum Perturbation Simulations on Exascale Supercomputers

Author – Zhikun Wu, Yangjun Wu, Ying Liu, Honghui Shang, Yingxiang Gao, Zhongcheng Zhang, Yuyang Zhang, Yingchi Long, Xiaobing Feng, Huimin Cui

Year – 2023

Description – Quantum perturbation theory is pivotal in determining the critical physical properties of materials. The first-principles computations of these properties have yielded profound and quantitative insights in diverse domains of chemistry and physics. In this work, we propose a portable and scalable OpenCL implementation for quantum perturbation theory, which can be generalized across various high-performance computing (HPC) systems. Optimal portability is realized through the utilization of a cross-platform unified interface and a collection of performance-portable heterogeneous optimizations. Exceptional scalability is attained by addressing major constraints on memory and communication, employing a locality-enhancing task mapping strategy and a packed hierarchical collective communication scheme. Experiments on two advanced supercomputers demonstrate that our implementation exhibits remarkably performance on various material systems, scaling the system to 200,000 atoms with all-electron precision. This research enables all-electron quantum perturbation simulations on substantially larger molecular scales, with a potentially significant impact on progress in material sciences.

Title – Trends in High Performance Computing: Exascale Systems and Facilities Beyond the First Wave

Author – Lynn A. Parnell, Dustin W. Demetriou, Vinod Kamath, Eric Y. Zhang

Year – 2019

Description – The demand for exascale computing is driving the development of high-performance systems worldwide. The United States, China, Japan, and Europe are all planning to deploy exascale systems in the next decade. These systems will feature a combination of traditional processors and co-processors like GPUs. The first exascale systems are expected to require massive resources, including 40 MW of power, 13,000 tons of liquid cooling, and 15,000 square feet of facility space. However, this is just the beginning, as the trend indicates that future systems will be more efficient and compact. To extend exascale computing beyond the first wave, a sustainable approach is needed, balancing custom hardware designs with operational costs. The migration to more direct liquid cooling approaches requires planning and standardization. Infrastructure compatibility across vendors will be crucial for the success of new cooling approaches in data centers. This paper discusses the challenges and opportunities for operators of second-wave exascale facilities in developing a data center strategy over the next decade.

Title – Workload Characterization for Exascale Computing Networks

Author – José Duro, Salvador Petit, Julio Sahuquillo, Maria E. Gomez

Year – 2018

Description - Exascale computing is the next step in high performance computing provided by systems composed of millions of interconnected processing cores. In order to guide the design and implementation of such systems, multiple workload characterization studies and system performance evaluations are required. This paper provides a workload characterization study in the context of the European Project ExaNeSt, which focuses, among others, on developing the network technology required to implement future exascale systems. In this work, we characterize different ExaNeSt applications from the computer network perspective by analyzing the distribution of messages, the dynamic bandwidth consumption, and the spatial communication patterns among cores. The analysis highlights three main observations; i) message sizes are, in general, below 50 kB; ii) communication patterns are usually bursty; and iii) spatial communication among cores concentrate in hot spots for most applications. Taking into account these observations, one can conclude that in order to unclog congested network links, an exascale network must be designed to briefly support higher-than-average bandwidths in the vicinity of key network cores.

Title – The Way to Develop Software Towards Exascale Computing

Author – Hao Li, Yuhua Tang, Xiaoguang Ren, Liyang Xu, Xinhai Xu

Year – 2016

Description - This paper preliminarily discusses how to develop software towards exascale computing. Two typical development models are studied, one is “all-round contract” which means all the functional modules are implemented in a single framework, and the other is “co-design” which using the existing packages to realize functional modules. As a widely used CFD software whose numerical solving module is implemented in a typical “all-round contract” manner, OpenFOAM is concerned in this work. After redesigning and reimplementing the solving module of OpenFOAM in a “co-design” way by inserting PETSc, the source lines of the code decrease dramatically, which improves the development efficiency. Tests of two CFD benchmark cases and a practical large-scale case on a 128-node cluster show that the newly implemented numerical module also has a higher solving efficiency, compared with the original numerical module in OpenFOAM. Therefore, it is recommended to develop software in a “co-design” manner towards exascale computing, and softwares already implemented in “all-round contract” way should also be reconsidered.

Title – Ponte Vecchio: A Multi-Tile 3D Stacked Processor for Exascale Computing

Author – Wilfred Gomes, Altug Koker, Pat Stover, Doug Ingerly, Scott Siers, Srikrishnan Venkataraman, Chris Pelto, Tejas Shah

Year – 2022

Description – Ponte Vecchio (PVC) is a heterogenous petaop 3D processor comprising 47 functional tiles on five process nodes. The tiles are connected with Foveros [1] and EMIB [2] to operate as a single monolithic implementation enabling a scalable class of Exascale supercomputers. The PVC design contains> 100B transistors and is composed of sixteen TSMC N5 compute tiles, and eight Intel 7 memory tiles optimized for random access bandwidth-optimized SRAM tiles (RAMBO) 3D stacked on two Intel 7 Foveros base dies. Eight HBM2E memory tiles and two TSMC N7 SerDes connectivity tiles are connected to the base dies with 11 dense embedded interconnect bridges (EMIB). SerDes connectivity provides a high-speed coherent unified fabric for scale-out connectivity between PVC SoCs. Each tile includes an 8-port switch enabling up to 8-way fully connected configuration supporting 90G SerDes links. The SerDes tile supports load/store, bulk data transfers and synchronization semantics that are critical for scale-up in HPC and AI applications. A 24-layer (11-2-11) substrate package houses the 3D Stacked Foveros Dies and EMIBs. To handle warpage, low-temperature solder (LTS) was used for Flip Chip Ball Grid Array (FCBGA) design for these die and package sizes.

Title – An Analysis of Resilience Techniques for Exascale Computing Platforms

Author – Daniel Dauwe, Sudeep Pasricha, Anthony A. Maciejewski, Howard Jay Siegel

Year – 2017

Description - With the increase in the complexity and number of nodes in large-scale high performance computing (HPC) systems, the probability of applications experiencing failures has increased significantly. As the computational demands of applications that execute on HPC systems increase, projections indicate that applications executing on exascale-sized systems are likely to operate with a mean time between failures (MTBF) of as little as a few minutes. A number of strategies for enabling fault resilience in systems of extreme sizes have been proposed in recent years. However, few studies provide performance comparisons for these resilience techniques. This work provides a comparison of four state-of-the-art HPC resilience techniques that are being considered for use in exascale systems. We explore the behavior of each resilience technique under simulated execution of a diverse set of applications varying in communication behavior and memory use. We examine how each resilience technique behaves as application size scales from what is considered large today through to exascale-sized applications. We further study the performance degradation that a large-scale system experiences from the overhead associated with each resilience technique as well as the application computation needed to continue execution when a failure occurs. Using the results from these analyses, we examine how application performance on exascale systems can be improved by allowing the system to select the optimal resilience technique for use in an application-specific manner, depending upon each application's execution characteristics.

CHAPTER 3

METHODOLOGY

3.1 PARALLEL ALGORITHMS -

Parallel algorithms are a fundamental component of exascale computing, enabling the efficient utilization of the massive parallelism available in modern supercomputers. These algorithms are designed to break down computational tasks into smaller, independent sub-tasks that can be executed simultaneously on multiple processing units, such as CPU cores or GPUs.

At the heart of parallel algorithms is the concept of parallelism, which allows multiple operations to be performed concurrently. This is in contrast to sequential algorithms, which execute one operation at a time. In exascale computing, where systems may contain millions of processing units, parallel algorithms are essential for achieving high performance and scalability.

There are several key techniques used in parallel algorithms:

Task Decomposition: This involves breaking down a large task into smaller, more manageable sub-tasks that can be executed independently. Each sub-task is then assigned to a different processing unit for execution.

Data Decomposition: In data decomposition, the dataset is divided into smaller subsets, and each subset is processed by a different processing unit. This is often used in problems where the data can be partitioned, such as in image processing or simulation.

Task Parallelism: Task parallelism involves executing multiple tasks concurrently. This can be achieved by assigning different tasks to different processing units or by using threading techniques to execute tasks in parallel within a single processing unit.

Data Parallelism: Data parallelism involves performing the same operation on multiple data elements concurrently. This is commonly used in array and matrix operations, where the same operation is applied to each element of the array or matrix.

Parallel algorithms must also consider issues such as load balancing, synchronization, and communication overhead. Load balancing ensures that the workload is evenly distributed among

processing units, while synchronization ensures that different parts of the algorithm remain coordinated. Communication overhead refers to the time and resources required to exchange data between processing units, which can impact performance.

Ex-

- **Parallel Matrix Multiplication:** In this algorithm, matrices are divided into smaller submatrices, and each submatrix multiplication is performed in parallel by different processing units. This is a classic example of data parallelism.
- **Parallel Sorting Algorithms:** Sorting large datasets can be computationally intensive. Parallel sorting algorithms, such as parallel quicksort or parallel mergesort, divide the dataset into smaller subsets and sort them in parallel before merging the results. This approach improves sorting performance for large datasets.
- **Parallel Breadth-First Search (BFS):** BFS is a fundamental graph traversal algorithm used in many applications, such as network analysis and pathfinding. In parallel BFS, each node in the graph is processed by a different processing unit, enabling efficient traversal of large graphs.
- **Parallel Monte Carlo Methods:** Monte Carlo methods are used for numerical simulation and optimization. Parallelizing Monte Carlo simulations involves running multiple simulations concurrently with different random number seeds to improve efficiency and reduce computation time.
- **Parallel Convolutional Neural Networks (CNNs):** CNNs are widely used in deep learning for image recognition and other tasks. Parallelizing CNN training involves distributing the computation of gradients and weight updates across multiple processing units to accelerate training.

3.2 MESSAGE PASSING INTERFACE –

Message Passing Interface (MPI) is a standardized and portable message-passing system designed to facilitate parallel computing on distributed memory architectures. It is a widely used programming model for writing parallel applications where multiple processes communicate with each other by sending and receiving messages.

Key features of MPI include:

Message Passing: MPI provides a set of functions for sending and receiving messages between processes. Processes can communicate in point-to-point or collective modes, allowing for efficient data exchange.

Process Management: MPI allows programmers to create, manage, and control multiple processes running concurrently. This enables the parallel execution of code across distributed memory systems.

Data Types: MPI supports various data types, including basic types (integers, floating-point numbers) and derived types (arrays, structures), allowing for flexible and efficient data communication.

Collective Operations: MPI provides collective operations that enable synchronization and data exchange among a group of processes. Examples include broadcast, reduce, scatter, and gather operations.

Fault Tolerance: Some MPI implementations offer features for fault tolerance, allowing applications to recover from process failures without terminating the entire program.

MPI is commonly used in scientific computing, simulations, and other high-performance computing applications where parallelism is essential. It provides a powerful and flexible programming model for leveraging the capabilities of modern parallel computing architectures.

3.3 OPEN MULTI-PROCESSING (OpenMP) –

OpenMP (Open Multi-Processing) is an API (Application Programming Interface) that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran. It allows developers to write parallel code that can run on multicore processors, shared-memory systems, and even distributed-memory systems with some limitations.

Key features of OpenMP include:

Pragmas: OpenMP uses compiler directives (pragmas) to parallelize code. These directives are hints to the compiler to parallelize the specified code blocks.

Shared Memory Model: OpenMP assumes a shared memory programming model, where all threads have access to a shared address space. This makes it easier to parallelize loops and sections of code.

Worksharing Constructs: OpenMP provides constructs like parallel, for, sections, and single to parallelize loops, sections, and single tasks.

Runtime Library: OpenMP includes a set of runtime library routines that manage thread creation, synchronization, and data sharing.

Portability: OpenMP is supported by most major compilers, making code written with OpenMP portable across different platforms.

Ease of Use: Compared to lower-level threading libraries like pthreads, OpenMP is often easier to use and requires fewer lines of code to parallelize a program.

Scalability: While OpenMP is excellent for parallelizing loops and simple parallel regions, it may not be as scalable or suitable for complex parallel algorithms as message passing interface (MPI).

Overall, OpenMP is a powerful tool for parallel programming, particularly for shared-memory systems and multicore processors, providing a balance between performance and ease of use.

3.4 OPENCL (OPEN COMPUTING LANGUAGE):

OpenCL is a framework for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, and other processors. It is maintained by the Khronos Group and provides a standard interface for parallel computing using task- and data-based parallelism. OpenCL allows developers to write code in a single-source style, where a single program can contain both the host code (executed on the CPU) and the device code (executed on the GPU or other accelerators).

3.5 CUDA (COMPUTE UNIFIED DEVICE ARCHITECTURE):

CUDA is a parallel computing platform and programming model developed by NVIDIA for their GPUs. It allows developers to use a C-like language to write programs (kernels) that run on NVIDIA GPUs. CUDA provides a comprehensive set of APIs for managing memory, launching kernels, and synchronizing between the CPU and GPU. It has gained popularity for its performance and ease of use in programming NVIDIA GPUs for general-purpose computing tasks.

Key Differences:

Platform Support: OpenCL is platform-independent and can be used on a variety of hardware from different vendors, including CPUs, GPUs, and accelerators. CUDA, on the other hand, is specific to NVIDIA GPUs and cannot be used on other hardware platforms.

Programming Model: OpenCL follows a more abstract and general-purpose programming model, making it suitable for a wide range of heterogeneous platforms. CUDA, being specific to NVIDIA GPUs, offers more direct control and optimization for NVIDIA GPU architectures.

Ecosystem: CUDA has a more mature and well-established ecosystem with extensive libraries and tools developed specifically for NVIDIA GPUs. OpenCL has a broader ecosystem with support from multiple vendors but may not have the same level of optimization and tooling for specific hardware platforms.

In summary, OpenCL offers a more flexible and vendor-agnostic approach to parallel computing across different hardware platforms, while CUDA provides a specialized and optimized solution for programming NVIDIA GPUs. The choice between them depends on factors such as hardware compatibility, performance requirements, and ecosystem support.

3.6 TASK BASED PARALLELISM -

Task-based parallelism is a programming paradigm where tasks, which represent units of work, are created and executed concurrently to improve the efficiency of a program. In this paradigm, instead of explicitly defining how tasks should be executed in a specific order, the programmer defines the tasks themselves and lets the underlying system manage their execution.

Key concepts of task-based parallelism include:

Task Creation: Tasks are created by the programmer to represent individual units of work. These tasks can be created dynamically based on the workload and dependencies between tasks.

Task Dependency: Tasks can have dependencies on other tasks, meaning that a task cannot start execution until its dependencies have been completed. This ensures that tasks are executed in the correct order to produce the correct result.

Task Scheduling: The system's runtime scheduler is responsible for scheduling tasks for execution on available processing resources. The scheduler decides when and where to execute each task based on factors such as resource availability and task dependencies.

Load Balancing: Task-based parallelism helps in achieving load balancing, as the scheduler can dynamically distribute tasks across available processors to ensure that all processors are utilized efficiently.

Scalability: Task-based parallelism can scale well to large numbers of processors, as the scheduler can distribute tasks across processors dynamically, adapting to changes in the system's workload and resource availability.

Task-based parallelism is particularly useful in applications where the workload is dynamic or where the dependencies between tasks are complex and not easily predictable. It allows for more flexible and efficient parallel execution compared to traditional approaches, where the programmer must explicitly manage the parallel execution of tasks.

Ex-

- **Intel Threading Building Blocks (TBB):** TBB is a C++ template library developed by Intel for task-based parallelism. It provides a rich set of features for creating and managing tasks, as well as algorithms for parallel execution.
- **Cilk/Cilk Plus:** Cilk is a programming language designed for multithreaded parallel computing. Cilk Plus is an extension of C and C++ that adds support for task parallelism. It provides keywords like cilk_spawn and cilk_sync for creating and synchronizing tasks.

- **Chapel:** Chapel is a parallel programming language developed by Cray. It provides built-in support for task parallelism, allowing programmers to express parallelism at a high level without needing to manage low-level details.

3.7 DATA MANAGEMENT AND STORAGE –

Data management and storage are critical aspects of exascale computing, where massive amounts of data need to be efficiently stored, accessed, and managed. Several key considerations and technologies play a role in data management and storage at this scale:

Scalability: Exascale systems must be able to scale storage capacity and performance to handle the vast amounts of data generated and processed. This scalability often involves distributed storage solutions that can scale out to thousands of nodes.

Parallel File Systems: Parallel file systems are designed to provide high-performance access to data across multiple storage devices and servers. They are essential for exascale computing to handle the large volume of data and the high concurrency of access.

Data Movement: Efficient data movement is crucial in exascale computing, both within the system and between different systems. High-speed interconnects and data transfer protocols are used to minimize latency and maximize throughput.

Data Replication and Backup: Given the critical nature of the data being processed, exascale systems often incorporate mechanisms for data replication and backup to ensure data durability and availability.

Data Organization and Indexing: Proper organization and indexing of data are essential for efficient data retrieval and analysis. Techniques such as data partitioning, indexing, and compression are used to optimize data access and storage.

Data Lifecycle Management: Managing the lifecycle of data, from creation to archival or deletion, is important in exascale computing to ensure that storage resources are used efficiently and data is retained as needed for analysis or compliance purposes.

Data Security and Privacy: With the increase in data volume and sensitivity, data security and privacy become paramount. Encryption, access controls, and other security measures are implemented to protect data from unauthorized access or loss.

Metadata Management: Metadata, which provides information about the data, is crucial for efficient data management and retrieval. Metadata management systems are used to store and manage metadata associated with the vast amount of data in exascale computing.

Exascale computing requires advanced data management and storage solutions to handle the massive volume of data generated and processed. Scalability, performance, efficiency, and security are key considerations in designing these solutions for exascale systems.

Ex-

- **Lustre:** Lustre is an open-source parallel distributed file system designed for large-scale cluster computing. It is known for its high performance and scalability, making it a popular choice for HPC environments.
- **IBM Spectrum Scale (formerly GPFS):** IBM Spectrum Scale is a high-performance, highly scalable file system used in HPC, big data, and cloud environments. It provides advanced features such as file striping, automatic data tiering, and high availability.
- **BeeGFS:** BeeGFS (formerly FhGFS) is a parallel file system developed specifically for high-performance computing and data-intensive workloads. It is known for its simplicity, scalability, and high performance.
- **CephFS:** CephFS is a distributed file system built on top of the Ceph distributed storage system. It provides a POSIX-compliant file system interface and is designed to be scalable and fault-tolerant.
- **IBM Elastic Storage System (ESS):** IBM ESS is a high-performance, scalable storage solution that includes IBM Spectrum Scale as the underlying parallel file system. It is designed for demanding HPC and AI workloads.
- **Panasas ActiveStor:** Panasas ActiveStor is a parallel file system designed for HPC and AI workloads. It offers high performance and scalability, with features such as parallel data access, automatic data tiering, and high availability.

- **OrangeFS:** OrangeFS is an open-source parallel file system designed for high-performance computing and big data applications. It is known for its scalability, reliability, and ease of use.

These parallel file systems are designed to provide high performance, scalability, and reliability for large-scale computing environments, making them suitable for exascale computing and other demanding workloads.

3.8 WORK FLOW MANAGEMENT SYSTEMS –

Workflow management systems (WMS) are software platforms designed to facilitate the automation, execution, and monitoring of workflows in various domains, including scientific research, business processes, and data analysis. These systems provide a framework for defining, scheduling, and executing a series of tasks or processes in a specified order to achieve a specific goal. Here are some key aspects of workflow management systems:

Workflow Definition: WMS allows users to define workflows visually or through a scripting language. Workflows typically consist of a series of interconnected tasks or steps, along with the dependencies between them.

Task Automation: WMS automates the execution of tasks within a workflow, reducing the need for manual intervention. Tasks can include data processing, analysis, file transfers, and more.

Dependency Management: WMS manages dependencies between tasks, ensuring that tasks are executed in the correct order based on their dependencies. This helps in maintaining the integrity and consistency of the workflow.

Resource Allocation: WMS allocates resources such as computing resources, storage, and software licenses based on the requirements of the workflow. This helps in optimizing resource utilization and improving overall efficiency.

Monitoring and Logging: WMS provides tools for monitoring the progress of workflows and individual tasks. It also logs relevant information such as task completion status, execution times, and any errors or exceptions that occur during execution.

Error Handling: WMS includes mechanisms for handling errors and exceptions that occur during workflow execution. This may involve retrying failed tasks, notifying users or administrators, or taking corrective actions automatically.

Integration: WMS often integrates with other software systems and tools, such as databases, cloud services, and scientific computing libraries, to enable seamless execution of workflows across different environments.

Scalability: WMS are designed to scale to handle large and complex workflows. They can manage workflows involving thousands of tasks running on distributed computing resources.

Examples of popular workflow management systems include Apache Airflow, Nextflow, Pegasus, and Galaxy. These systems are widely used in scientific research, data analysis, and business process automation to streamline and optimize workflow execution.

Ex-

- **Apache Airflow:** Apache Airflow is an open-source platform for creating, scheduling, and monitoring workflows. It allows users to define workflows as Directed Acyclic Graphs (DAGs) and provides a rich set of features for managing dependencies, executing tasks, and handling retries and failures.
- **Nextflow:** Nextflow is a bioinformatics workflow management system that enables the creation and execution of scalable and reproducible workflows. It supports parallel and distributed computing environments and provides robust support for data-driven workflows in genomics and other life sciences.
- **Pegasus**: Pegasus is a workflow management system designed for large-scale scientific workflows. It enables users to define complex workflows with dependencies and provides mechanisms for managing data movement, job execution, and provenance tracking in distributed computing environments.
- **Galaxy:** Galaxy is an open-source platform for data-intensive biomedical research. It provides a web-based interface for creating and running workflows, enabling researchers to analyze and share data in a reproducible manner.

- **Taverna:** Taverna is a workflow management system designed for scientific workflows in bioinformatics, chemistry, and other domains. It allows users to create workflows using a visual interface and provides tools for integrating with external services and databases.
- **KNIME:** KNIME is a platform for creating and executing data science workflows. It provides a visual workflow editor and a wide range of plugins for data integration, processing, analysis, and visualization.
- **Luigi:** Luigi is a Python-based workflow management system developed by Spotify. It allows users to define workflows as Python classes and provides tools for scheduling, executing, and monitoring tasks.

3.9 PERFORMANCE OPTIMIZATION TOOLS –

Performance optimization tools are software tools used to improve the performance of computer systems, applications, and processes. These tools analyze various aspects of the system or application and suggest or implement optimizations to enhance performance. Here are some common types of performance optimization tools:

Profiling Tools: Profiling tools are used to analyze the runtime behavior of a program. They gather information about the program's execution, such as the time spent in each function, memory usage, and I/O operations. This information can help identify bottlenecks and areas for optimization.

Code Analysis Tools: Code analysis tools examine source code to identify potential performance issues, such as inefficient algorithms, redundant code, or memory leaks. These tools can provide recommendations for improving code efficiency and performance.

Compiler Optimizations: Compilers often include optimization options that can improve the performance of compiled code. These optimizations can include loop unrolling, inlining functions, and optimizing memory access patterns.

Profiling and Monitoring Tools: Profiling and monitoring tools provide real-time performance data about a system or application. They can help identify performance bottlenecks and track the impact of optimizations over time.

Memory Profilers: Memory profilers are used to analyze an application's memory usage. They can help identify memory leaks, inefficient memory allocation patterns, and ways to optimize memory usage.

Parallelization Tools: Parallelization tools help developers parallelize their code to take advantage of multicore processors and distributed computing environments. These tools can automate the process of parallelizing code and optimize its performance.

Database Performance Tools: For applications that interact with databases, database performance tools can help optimize database queries, indexes, and overall database performance.

Network Performance Tools: Network performance tools analyze and optimize network traffic to improve the performance of networked applications.

Load Testing Tools: Load testing tools simulate high levels of user activity to test the performance and scalability of an application under load. These tools can help identify performance bottlenecks and optimize the application's performance under heavy loads.

Profiling and Tracing Tools: Profiling and tracing tools provide detailed information about the execution of a program, including function calls, resource usage, and timing information. These tools can help diagnose performance issues and optimize code for better performance.

Overall, performance optimization tools play a crucial role in ensuring that computer systems and applications perform efficiently and effectively, meeting the performance requirements of users and applications.

Ex-

Profiling Tools:

- **gprof:** A profiling tool for Unix-like systems that can analyze the execution time of different parts of a program.
- **Intel VTune Profiler:** Provides performance profiling for C, C++, Fortran, Python, and Java applications.

Code Analysis Tools:

- **Valgrind:** A suite of tools for debugging and profiling that includes tools for memory profiling and cache profiling.
- **Clang Static Analyzer:** A static analysis tool that can identify bugs and performance issues in C, C++, and Objective-C code.

Compiler Optimizations:

- **GCC Compiler:** The GNU Compiler Collection includes optimizations for improving code performance.
- **LLVM Compiler:** The LLVM compiler framework includes optimizations for improving code performance and supports multiple programming languages.

Profiling and Monitoring Tools:

- **perf:** A profiling tool for Linux that provides detailed performance data about system and application behavior.
- **DTrace:** A dynamic tracing framework for analyzing the behavior of software in real time.
- Memory Profilers:
- **Valgrind Memcheck:** A memory error detector that can detect memory leaks and other memory-related errors.
- **Heaptrack:** A memory profiler for Linux that can track memory allocations and deallocations in C and C++ programs.

Parallelization Tools:

- **OpenMP:** A set of compiler directives and library routines for parallel programming in C, C++, and Fortran.
- **CUDA Toolkit:** A parallel computing platform and programming model developed by NVIDIA for GPUs.

Database Performance Tools:

- **MySQL Performance Schema:** A feature for MySQL that provides performance data about database operations.
- **Oracle SQL Tuning Advisor:** A tool for analyzing SQL statements and providing recommendations for improving performance.

Network Performance Tools:

- **Wireshark:** A network protocol analyzer that can capture and analyze network traffic to identify performance issues.
- **iperf:** A tool for measuring TCP and UDP bandwidth performance between two hosts.
- Load Testing Tools:
- **Apache JMeter**: A Java-based tool for load testing web applications and services.
- **Gatling:** A Scala-based load testing tool for web applications.

Profiling and Tracing Tools:

- **strace:** A system call tracer for Linux that can trace system calls and signals.
- **DTrace:** A dynamic tracing framework for analyzing the behavior of software in real time, also available on Linux with the BPF Compiler Collection (BCC).

3.10 RESOURCE MANAGEMENT AND SCHEDULING –

Resource management and scheduling are critical aspects of high-performance computing (HPC) and exascale computing, where efficient utilization of resources is key to maximizing performance. Here's an overview of resource management and scheduling in the context of HPC:

Resource Management: Resource management involves the allocation and monitoring of various computing resources, such as processors, memory, storage, and network bandwidth, to different tasks or jobs in a computing environment. It ensures that resources are used efficiently and fairly among users or applications.

Job Scheduling: Job scheduling is the process of deciding when and where to execute tasks or jobs on the available resources. It involves selecting the best resources for a job based on factors such as resource availability, job requirements, and system policies. Schedulers aim to maximize system utilization, minimize job waiting times, and ensure fairness among users.

Batch Scheduling: In HPC environments, batch scheduling is commonly used, where jobs are submitted to a queue and scheduled to run when resources become available. Batch schedulers prioritize jobs based on criteria such as job size, priority, and resource requirements.

Interactive Scheduling: Some systems also support interactive scheduling, where users can request immediate access to resources for interactive tasks. Interactive scheduling requires faster response times and often involves different scheduling policies than batch scheduling.

Policy-Based Scheduling: Schedulers use policies to make decisions about resource allocation and job scheduling. Policies can be based on factors such as job priority, resource availability, fairness, and performance goals. Different systems may have different policies depending on their specific requirements and constraints.

Dynamic Scheduling: In dynamic scheduling, jobs are scheduled based on real-time conditions, such as resource availability and workload changes. Dynamic schedulers can adapt to changing conditions to optimize resource utilization and job performance.

Resource Reservation: Some systems support resource reservation, where users can reserve resources in advance for their jobs. Reservation systems ensure that resources are available when needed and can help guarantee job deadlines are met.

Fault Tolerance: Resource management systems often include mechanisms for handling system failures and ensuring job continuity. This may involve checkpointing, job migration, or rescheduling to other resources.

Optimization: Resource management and scheduling systems aim to optimize various aspects of job execution, including minimizing job turnaround time, maximizing system utilization, and ensuring fairness among users.

Resource management and scheduling are complex tasks in HPC and exascale computing, requiring sophisticated algorithms and policies to handle the scale and complexity of modern computing environments. Efficient resource management and scheduling are essential for achieving high performance and productivity in HPC systems.

Ex-

- **SLURM (Simple Linux Utility for Resource Management):** SLURM is a widely used open-source workload manager and job scheduler for Linux clusters. It supports both batch and interactive job scheduling and provides a flexible and scalable framework for managing resources.

- **PBS (Portable Batch System):** PBS is a popular workload management system that provides job scheduling and resource management capabilities for HPC clusters. It allows users to submit, monitor, and control jobs on a cluster of computers.
- **Moab Workload Manager:** Moab is a commercial workload manager and job scheduler that provides advanced scheduling and policy-based management capabilities. It is often used in conjunction with the TORQUE resource manager.
- **IBM Spectrum LSF (Load Sharing Facility):** LSF is a comprehensive workload management solution for HPC environments, providing job scheduling, resource management, and workload analytics capabilities. It is designed to optimize resource utilization and improve job throughput.
- **Grid Engine:** Grid Engine is an open-source batch queuing system that provides job scheduling and resource management for distributed computing environments. It supports a wide range of applications and workload types.
- **OpenPBS:** OpenPBS (Open Portable Batch System) is an open-source version of the PBS workload management system. It provides job scheduling and resource management capabilities for HPC clusters and supercomputers.

These systems offer varying levels of functionality and scalability, allowing organizations to choose the one that best meets their needs for managing resources and scheduling jobs in HPC and exascale computing environments.

3.11 MACHINE LEARNING AND AI

Machine learning (ML) and artificial intelligence (AI) are transformative technologies that enable computers to learn from data and perform tasks that traditionally required human intelligence. ML is a subset of AI that focuses on developing algorithms that can learn from and make predictions or decisions based on data. AI, on the other hand, encompasses a broader range of technologies and approaches aimed at mimicking human cognitive functions such as reasoning, learning, and problem-solving.

ML algorithms can be broadly categorized into three types: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the algorithm learns from labeled training data, where each data point is associated with a label or outcome. Unsupervised learning involves learning patterns and structures from unlabeled data, while reinforcement learning learns through a trial-and-error process by interacting with an environment and receiving feedback.

AI systems can be applied across various domains, including healthcare, finance, transportation, and more. For example, in healthcare, AI can be used to analyze medical images for disease detection, assist in diagnosis and treatment planning, and personalize patient care. In finance, AI can be used for fraud detection, algorithmic trading, and customer service automation.

Recent advances in deep learning, a subset of ML that uses neural networks with many layers, have significantly improved the performance of AI systems in tasks such as image and speech recognition, natural language processing, and autonomous driving. These advances have fueled the rapid adoption of AI and ML in various industries, leading to the development of new products, services, and business models.

Ex-

Supervised Learning Algorithms:

- Linear Regression
- Logistic Regression
- Support Vector Machines (SVM)
- Decision Trees
- Random Forests
- Gradient Boosting Machines (GBM)
- Neural Networks

Unsupervised Learning Algorithms:

- K-Means Clustering
- Hierarchical Clustering
- Principal Component Analysis (PCA)
- Independent Component Analysis (ICA)

- t-Distributed Stochastic Neighbor Embedding (t-SNE)
- Generative Adversarial Networks (GANs)
- **Reinforcement Learning Algorithms:**
- Q-Learning
- Deep Q Networks (DQN)
- Policy Gradient Methods
- Actor-Critic Methods

Natural Language Processing (NLP):

- Word Embeddings (e.g., Word2Vec, GloVe)
- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory (LSTM) Networks
- Transformer Models (e.g., BERT, GPT)

Computer Vision:

- Convolutional Neural Networks (CNNs)
- Region-Based CNNs (R-CNN, Fast R-CNN, Faster R-CNN)
- Single Shot MultiBox Detector (SSD)
- You Only Look Once (YOLO)

Recommendation Systems:

- Collaborative Filtering
- Content-Based Filtering
- Matrix Factorization Techniques (e.g., Singular Value Decomposition, Alternating Least Squares)

Deep Reinforcement Learning:

- Deep Q Networks (DQN)
- Deep Deterministic Policy Gradient (DDPG)
- Proximal Policy Optimization (PPO)
- Asynchronous Advantage Actor-Critic (A3C)

CHAPTER 4

APPLICATIONS

4.1 APPLICATIONS

1. **Weather Forecasting and Climate Modeling:** Exascale computing revolutionizes weather forecasting and climate modeling by enabling simulations at unprecedented resolutions, improving accuracy and predictability for better disaster preparedness and climate change mitigation.
2. **Astrophysics and Cosmology:** Exascale computing transforms our understanding of the universe by simulating complex astrophysical processes, such as galaxy formation and black hole dynamics, allowing researchers to test theories and make new discoveries.
3. **Drug Discovery and Molecular Modeling:** Exascale computing accelerates drug discovery by simulating molecular interactions and drug binding, enabling researchers to screen vast chemical libraries and identify potential drug candidates with greater speed and accuracy.
4. **Materials Science and Engineering:** Exascale computing revolutionizes materials science by simulating materials at the atomic and molecular level, facilitating the design of new materials with specific properties for applications in energy, electronics, and healthcare.
5. **Energy Exploration and Production:** Exascale computing transforms the energy sector by enabling more accurate reservoir modeling, seismic imaging, and simulation of complex geological formations, leading to improved exploration and production efficiency.
6. **Genomics and Personalized Medicine:** Exascale computing revolutionizes genomics and personalized medicine by analyzing large-scale genomic data to understand the genetic basis of diseases and develop personalized treatment strategies based on individual genetic profiles.
7. **Aerospace and Automotive Design:** Exascale computing transforms aerospace and automotive design by enabling high-fidelity simulations of aerodynamics, structural mechanics, and crash dynamics, leading to the development of safer and more efficient aircraft and vehicles.

8. **Financial Modeling and Risk Analysis:** Exascale computing revolutionizes financial modeling and risk analysis by simulating complex financial systems and market scenarios, enabling better risk management and investment strategies.
9. **Urban Planning and Traffic Management:** Exascale computing transforms urban planning and traffic management by simulating urban growth, traffic patterns, and transportation systems, leading to more efficient and sustainable urban development.
10. **Artificial Intelligence and Machine Learning:** Exascale computing accelerates AI and machine learning by providing the computational power needed to train large-scale models and analyze massive datasets, leading to breakthroughs in AI research and applications.
11. **Scientific Discovery:** Exascale computing revolutionizes scientific research by providing unprecedented computational power for simulating complex phenomena, enabling researchers to explore new frontiers in physics, chemistry, and biology.
12. **Healthcare Innovation:** Exascale computing transforms healthcare by facilitating personalized medicine through detailed simulations of genetic interactions and disease pathways, leading to more effective treatments and improved patient outcomes.
13. **Engineering Design:** Exascale computing revolutionizes engineering design by enabling high-fidelity simulations of structures, materials, and systems, allowing engineers to optimize designs for safety, efficiency, and performance.
14. Climate Change Mitigation: Exascale computing plays a crucial role in addressing climate change by providing advanced modeling and simulation capabilities for predicting climate patterns, assessing risks, and developing mitigation strategies.
15. **Natural Disaster Prediction**: Exascale computing enhances natural disaster prediction by simulating the behavior of complex systems such as weather patterns, earthquakes, and tsunamis, enabling early warning systems and better disaster preparedness.
16. **Energy Efficiency:** Exascale computing drives innovations in energy efficiency by simulating energy systems and optimizing energy production, distribution, and consumption, leading to reduced carbon emissions and sustainable energy solutions.
17. **Cybersecurity:** Exascale computing strengthens cybersecurity by enabling real-time analysis of vast amounts of data to detect and respond to cyber threats, enhancing the resilience of critical infrastructure and systems.

18. **Space Exploration:** Exascale computing advances space exploration by simulating spacecraft trajectories, planetary environments, and celestial phenomena, aiding in mission planning and the search for extraterrestrial life.
19. **Manufacturing Optimization:** Exascale computing optimizes manufacturing processes by simulating production lines, materials, and supply chains, leading to increased efficiency, reduced costs, and improved product quality.
20. **Data Analytics:** Exascale computing accelerates data analytics by processing massive datasets in real-time, uncovering valuable insights that drive business innovation and decision-making.
21. **Materials Science:** Exascale computing enables precise simulations of atomic and molecular interactions, facilitating the discovery of new materials with specific properties, such as superconductors or lightweight alloys for aerospace applications.
22. **Genomics and Personalized Medicine:** Exascale computing accelerates genomic analysis and personalized medicine by analyzing vast amounts of genetic data to identify disease markers, develop targeted therapies, and predict individual health risks.
23. **Earth System Modeling:** Exascale computing enhances Earth system modeling by simulating climate dynamics, ocean circulation, and atmospheric processes at unprecedented resolutions, improving our understanding of climate change and its impacts.
24. **Financial Modeling:** Exascale computing revolutionizes financial modeling by enabling high-speed analysis of market data, risk assessment, and complex trading algorithms, enhancing decision-making in financial markets.
25. **Artificial Intelligence and Machine Learning:** Exascale computing powers advanced artificial intelligence (AI) and machine learning (ML) algorithms by processing massive datasets to train models for image recognition, natural language processing, and autonomous systems.
26. **Smart Cities and Urban Planning:** Exascale computing supports smart city initiatives by analyzing vast amounts of data from sensors, traffic patterns, and energy usage to optimize urban infrastructure, transportation systems, and resource management.

CHAPTER 5

ADVANTAGES & DISADVANTAGES

5.1 ADVANTAGES

- **Increased Parallelism:** Exascale computing enables a higher level of parallelism, allowing for more tasks to be executed simultaneously, thus improving overall performance.
- **Improved Efficiency**: Exascale systems are designed to be more energy-efficient, delivering higher computational power per watt.
- **Advanced Hardware Architectures:** Exascale systems incorporate innovative hardware designs, such as system-on-chip (SoC) and advanced memory technologies, to enhance performance and efficiency.
- **Enhanced Memory and Storage:** Exascale systems utilize high-bandwidth memory (HBM), non-volatile memory express (NVMe), and 3D-stacked memory to improve data access speeds and efficiency.
- **Optimized Interconnects:** Exascale systems use high-bandwidth interconnects, such as Infinity Fabric, to facilitate fast data transfer between components, reducing latency.
- **Massively Parallel Processing:** Exascale systems leverage GPU computing and technologies like CUDA and Tensor Cores to achieve massive parallel processing capabilities, improving performance for parallelizable tasks.
- **Software Optimization:** Exascale computing involves optimizing software for efficient use of hardware resources, enabling better performance and scalability.
- **Reduced Latency:** Exascale systems minimize latency in data transfer and computation, leading to faster processing and response times.
- **Improved Reliability:** Exascale systems incorporate advanced error detection and correction mechanisms to ensure reliable operation, even at extreme scales.
- **Scalability:** Exascale computing offers scalability, allowing systems to handle increasing computational demands and data volumes effectively.

- **Increased Innovation:** Exascale computing fosters innovation by providing researchers, engineers, and developers with the computational power to explore new ideas, algorithms, and technologies that can drive advancements in various fields.
- **Global Collaboration:** Exascale computing facilitates global collaboration by enabling researchers from around the world to work together on large-scale projects, sharing resources and expertise to address complex challenges that require massive computational resources.
- **Improved Prediction and Forecasting:** Exascale computing enables more accurate prediction and forecasting in various domains, such as weather forecasting, financial modeling, and risk analysis, leading to better decision-making.
- **Enhanced Data Analytics:** Exascale computing enhances data analytics capabilities, allowing for faster processing and analysis of large datasets, leading to deeper insights and better-informed decisions.
- **Optimized Resource Utilization:** Exascale computing helps optimize resource utilization in industries such as manufacturing, logistics, and energy, leading to increased efficiency and cost savings.
- **Accelerated Scientific Discovery**: Exascale computing accelerates scientific discovery by enabling researchers to simulate and analyze complex phenomena, leading to new insights and discoveries in fields such as physics, chemistry, and biology.
- **Improved Product Development:** Exascale computing facilitates faster and more accurate product development through simulations and modeling, reducing time to market and improving product quality.
- **Enhanced Security:** Exascale computing enables more robust security measures, such as advanced encryption and threat detection, enhancing cybersecurity in an increasingly digital world.
- **Empowered AI and Machine Learning:** Exascale computing empowers AI and machine learning applications by providing the computational power needed to train complex models and analyze large datasets, leading to more advanced AI solutions.

5.2 DISADVANTAGES

- **High Cost:** Building and maintaining exascale computing systems is expensive, requiring significant investment in hardware, software, and infrastructure.
- **Complexity:** Exascale computing systems are highly complex, requiring specialized expertise for design, development, and operation.
- **Data Security Risks:** Exascale computing introduces new challenges for data security and privacy, requiring robust security measures.
- **Software Challenges:** Developing software for exascale systems is challenging, requiring new algorithms and programming models.
- **Energy Consumption:** Exascale computing systems consume a large amount of energy, raising environmental concerns.
- **Cooling Requirements:** Exascale systems generate a significant amount of heat, requiring advanced cooling solutions.
- **Data Storage Challenges:** Managing and storing the massive amounts of data generated by exascale systems is a challenge.
- **Compatibility Issues:** Ensuring compatibility and interoperability with existing systems and software can be challenging.
- **Maintenance Complexity:** Exascale systems require regular maintenance and upgrades, adding to operational complexity.
- **Limited Accessibility:** Due to cost and complexity, exascale computing may be inaccessible to smaller organizations and researchers.
- **Programming Complexity:** Programming for exascale systems requires specialized skills and tools, making it challenging for developers to optimize code for performance and scalability.
- **Data Movement:** Exascale systems face challenges related to data movement, as moving large volumes of data between processors and memory can lead to latency and bottlenecks.

- **Resource Allocation:** Managing resources such as memory, storage, and processing power in exascale systems can be challenging, requiring careful planning and optimization to avoid wastage.
- **Data Privacy Concerns:** Exascale computing raises concerns about data privacy, as the massive computational power can potentially be used to analyze and exploit personal data without consent.
- **Ethical Dilemmas:** Exascale computing raises ethical dilemmas related to the use of advanced technologies for purposes that may harm individuals or society.
- **Digital Divide:** Exascale computing can widen the digital divide, as access to such advanced technology may be limited to certain groups or countries, creating inequalities in knowledge and opportunities.
- **Dependency on Technology**: Exascale computing can lead to a dependency on technology, making societies more vulnerable to disruptions and failures in the computing infrastructure.
- **Environmental Impact:** Exascale computing consumes a large amount of energy, leading to increased carbon emissions and environmental impact, especially if the energy source is not renewable.
- **Resource Competition**: Exascale computing may lead to competition for resources, such as rare earth elements used in hardware components, which can have environmental and geopolitical implications.
- **Job Displacement:** Exascale computing can lead to job displacement, as automation and AI may replace certain tasks and roles, leading to unemployment in some sectors.
- **Security Risks:** Exascale computing poses security risks, such as cyberattacks and data breaches, due to the increased complexity and interconnectedness of systems.
- **Regulatory Challenges:** Exascale computing raises regulatory challenges, as governments may struggle to keep pace with the rapid advancements in technology and its implications for society.
- **Cultural Impact:** Exascale computing can have a cultural impact, as society adapts to new ways of thinking and interacting with technology, leading to changes in behavior and norms.

CHAPTER 6

PROCEDURE

6.1 PROCEDURE

6.1.1 SEMICONDUCTOR FABRICATION PROCESS

Reduction in Process Nodes:

Continuously shrink process nodes to achieve higher transistor density and performance.

Address challenges such as increased leakage currents, variability, and manufacturing complexity.

Materials and Manufacturing Techniques:

Develop new materials and techniques to overcome physical limits of silicon-based transistors.

Explore alternative materials like III-V semiconductors and 2D materials (e.g., graphene).

Implement advanced manufacturing processes like extreme ultraviolet lithography (EUV) for finer feature sizes.

Heat Dissipation:

Implement efficient heat dissipation solutions to prevent overheating and maintain optimal performance.

Use advanced cooling techniques such as liquid cooling and innovative thermal management methods.

Power Consumption:

Develop low-power designs to mitigate increased power consumption with higher transistor density and performance.

Optimize power management strategies to improve energy efficiency and battery life for mobile devices.

Quantum Effects:

Address quantum effects that become more pronounced at smaller process nodes.

Mitigate issues such as electron tunneling and quantum confinement through novel design techniques and materials.

Design Complexity:

Utilize sophisticated design tools and methodologies to manage the increasing complexity of chip design and manufacturing.

Economic Viability:

Balance the costs associated with developing semiconductor technologies for smaller process nodes with potential benefits.

Environmental Impact:

Develop environmentally friendly manufacturing techniques to mitigate the environmental impact of semiconductor manufacturing processes.

This procedure outlines the steps needed to overcome scaling challenges in exascale computing, including major R&D progress, power management, system memory optimization, system resiliency, fault tolerance, programming models, I/O devices, tools development, thermal management, and silicon fabrication advancements.

Major R&D Progress:

Allocate significant resources for research and development to address scaling challenges.

Foster collaboration between industry, academia, and government to accelerate progress.

Power Challenge:

Develop and adopt energy-efficient components and architectures.

Implement power management techniques at both hardware and software levels.

System Memory Challenge:

Utilize high-bandwidth memory (HBM) and non-volatile memory technologies.

Implement advanced caching and data management strategies to optimize memory usage.

System Resiliency Challenge:

Design systems with built-in redundancy and fault tolerance mechanisms.

Implement error detection and correction codes (ECC) for memory and data integrity.

Fault Tolerance:

Develop and deploy fault-tolerant algorithms and software.

Use redundancy and error correction techniques to mitigate the impact of hardware failures.

Programming Models:

Develop and standardize programming models that support massive parallelism.

Provide tools and libraries to simplify programming for exascale systems.

I/O Devices:

Use high-speed interconnects and storage devices to meet I/O requirements.

Implement data compression and caching techniques to reduce I/O bottlenecks.

Tools:

Develop and deploy tools for performance monitoring, debugging, and optimization.

Provide comprehensive development environments to support exascale application development.

Thermal Management:

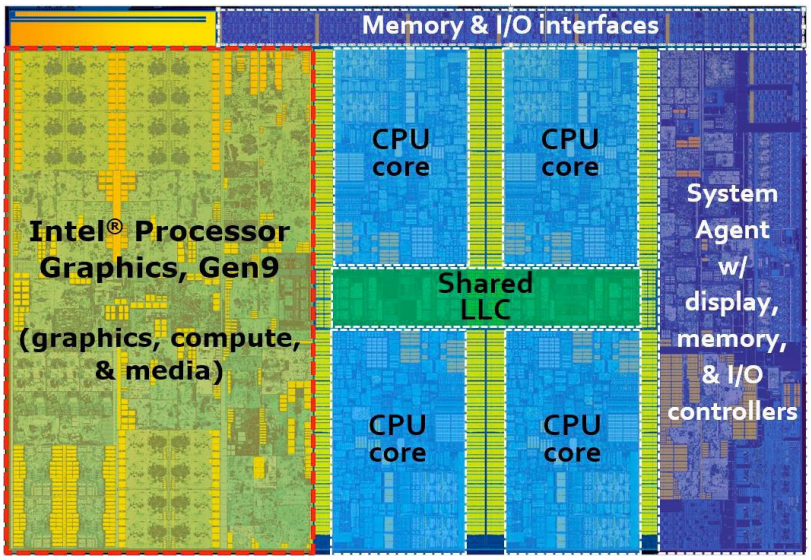
Design systems with efficient cooling solutions, such as liquid cooling and phase-change materials.

Monitor and manage system temperatures to prevent overheating and ensure reliability.

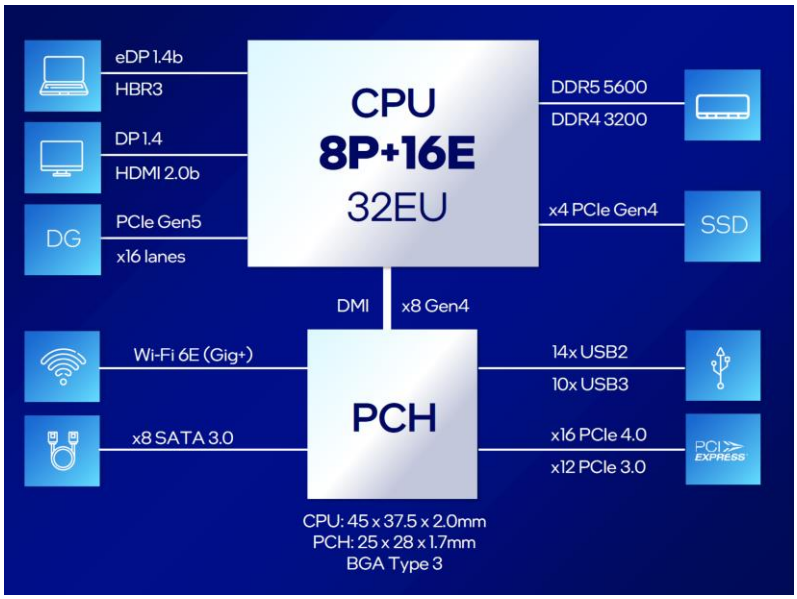
Silicon Fabrication:

Continue to advance semiconductor manufacturing processes to enable smaller process nodes.

Explore alternative materials and technologies to overcome the physical limits of silicon.



(Figure: Intel MonoLithic Architecture Alder Lake)



(Figure: Intel Meteor Lake Specs Core i9-14900KS)

6.1.2 IMPLEMENTING OF ALGORITHMS

New Multicore-Friendly and Multicore-Aware Algorithms:

Develop algorithms that can efficiently utilize the parallelism offered by multicore processors.

Design algorithms that are aware of the architecture of multicore systems to optimize performance.

Adaptive Response to Load Imbalance:

Implement algorithms that can dynamically adjust to uneven workload distribution across cores or nodes.

Use load balancing techniques to ensure efficient utilization of resources.

Multiple Precision Algorithms/Software:

Develop algorithms that can handle different precision requirements efficiently.

Implement software that can switch between precision levels based on computational needs.

Scheduling and Memory Management for Heterogeneity and Scale:

Design scheduling algorithms that can handle the diverse hardware configurations and scale of exascale systems.

Develop memory management techniques that can efficiently utilize the memory hierarchy in heterogeneous systems.

Auto-Tuning:

Create algorithms that can automatically adjust their parameters based on the characteristics of the system and workload.

Implement auto-tuning frameworks that can optimize performance without manual intervention.

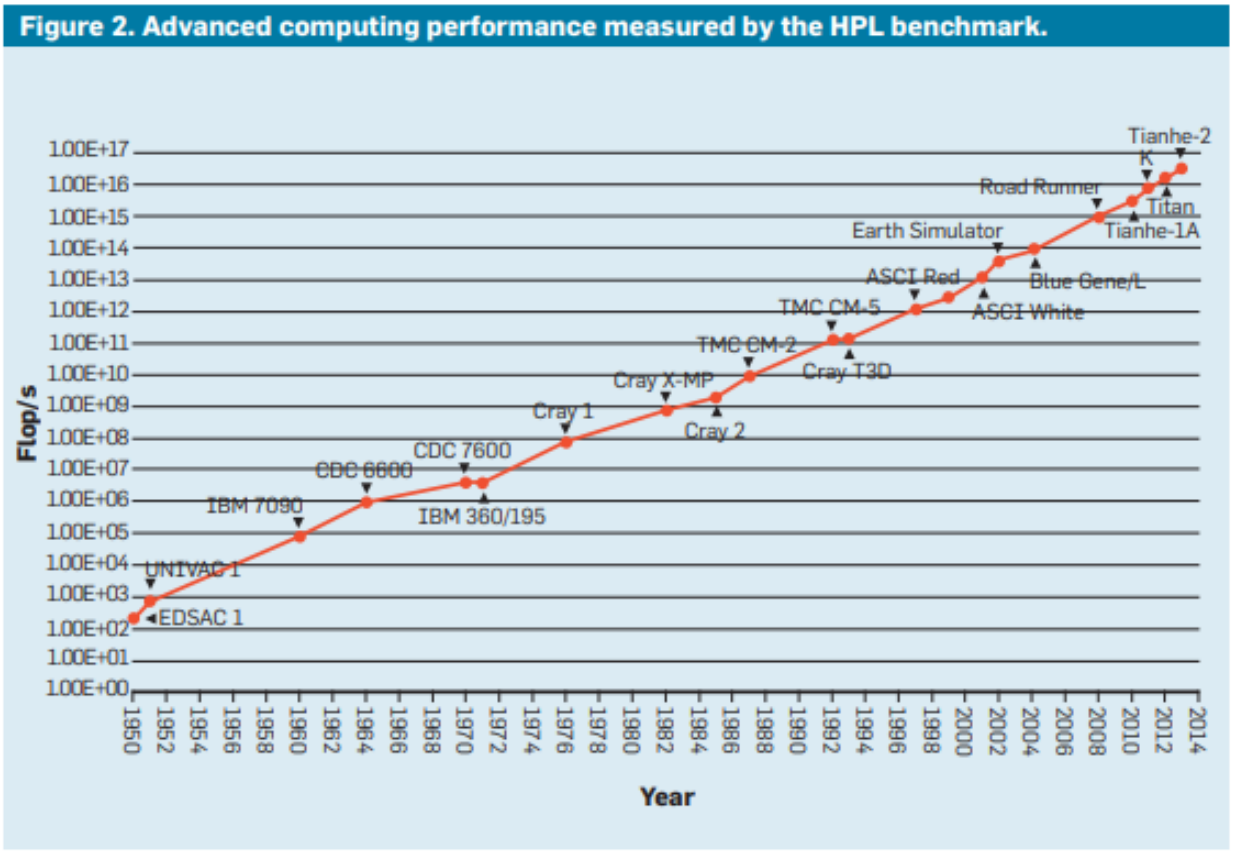
Communication Avoiding (Independence):

Design algorithms that minimize communication between cores or nodes.

Ensure that algorithms are data-independent whenever possible to reduce the need for communication.

It is essential to design algorithms that minimize communication and are adaptable to varying workloads and hardware configurations to achieve optimal performance in exascale computing.

6.1.3 PERFORMANCE ANALYSIS AND EVALUATION



(Figure: HPL Benchmark)

	2010	2018	Factor Change
System peak	2 Pf/s	1 Ef/s	500
Power	6 MW	20 MW	3
System Memory	0.3 PB	10 PB	33
Node Performance	0.125 Gf/s	10 Tf/s	80
Node Memory BW	25 GB/s	400 GB/s	16
Node Concurrency	12 cpus	1,000 cpus	83
Interconnect BW	1.5 GB/s	50 GB/s	33
System Size (nodes)	20 K nodes	1 M nodes	50
Total Concurrency	225 K	1 B	4,444
Storage	15 PB	300 PB	20
Input/Output bandwidth	0.2 TB/s	20 TB/s	100

Table 1: Potential Exascale Computer Design for 2018 and its relationship to current HPC designs. [59]

High-Performance Linpack (HPL): Exascale systems are expected to achieve HPL scores in the exaFLOP range, meaning they can perform at least one exaFLOP (10^18 floating-point operations per second) on this benchmark.

Graph500: This benchmark measures the performance of data-intensive graph algorithms. Exascale systems are expected to significantly outperform current supercomputers on this benchmark due to their massive parallel processing capabilities and memory bandwidth.

HPCG (High-Performance Conjugate Gradient): This benchmark is designed to complement HPL by measuring the performance of sparse matrix computations, which are common in many scientific simulations. Exascale systems are expected to excel in this benchmark as well.

HPL-AI: This benchmark is designed to evaluate the performance of AI workloads on supercomputers. It measures the performance of training deep learning models using large datasets. Exascale systems are expected to show significant improvements in AI workload performance compared to current supercomputers.

HACC: The Hardware/Hybrid Accelerated Cosmology Code (HACC) is a benchmark used to simulate the large-scale structure of the universe. Exascale systems are expected to enable more detailed and accurate simulations of cosmic structures, leading to new insights into the evolution of the universe.

6.2 HARFWARE REQUIREMENTS

High-Performance Processors: Exascale systems require processors capable of delivering high computational power, often featuring multi-core architectures with specialized co-processors like GPUs or accelerators.

- **Ex-** AMD EPYC 9654 ‘Genoa’ (96 Cores, 192 Threads, 3.7 GHz Turbo Frequency, 5nm Zen 4 Architecture, 12 Lane DDR5 or ECC Memory Upto 4800Mt/s with a 460.8 GB/s Bandwidth, PCIe 5.0 x128, Upto 400W TDP, 384MB L3 Cache)

Graphics Processing Unit (GPU): For Exascale computing GPUs play a crucial role in accelerating computations due to their parallel processing capabilities. They are used alongside traditional CPUs to offload parallelizable tasks, such as scientific simulations, machine learning, and data analytics, thereby significantly enhancing overall computational performance.

- **Ex-** NVIDIA H100 NVL1 with NVLink (Based on Hopper Architecture, 12592 Shading units, 456 Tensor Cores, Upto 67 TFLOPS for 64bit Double Floating Point Calculations)

Memory: Large memory capacity and high-bandwidth memory (HBM) are essential for handling the massive datasets and computations in exascale computing.

- **Ex-** SkHynix HBM3E or Micron HBM3E with Upto 1 TB/s Bandwidth

Interconnects: High-speed, low-latency interconnects such as InfiniBand or Ethernet are necessary for efficient communication between processors and nodes in exascale systems.

- **Ex-** Mellanox 800GbE or Cisco 1.6 TbE Ethernet standards

Storage: Exascale systems require large-scale storage solutions with high-speed access to accommodate the immense amount of data generated and processed.

- **Ex-** PCIe 5.0 NVMe 2.0 Kioxia CDP8-R with upto 128GT/s and upto 30TB Storage Space

Cooling Systems: Due to the high power consumption of exascale systems, efficient cooling solutions such as liquid cooling or advanced air cooling are essential to maintain optimal operating temperatures.

- **Ex-** CoolIT Systems DCLC (Direct Contact Liquid Cooling) or GRC Immersion Cooling with upto 5000W of Heat dissipation per 1U space.

High-Bandwidth Interconnects: Exascale systems require advanced interconnect technologies like InfiniBand or Intel's Omni-Path Architecture to ensure fast and efficient communication between nodes.

- **Ex-** InfiBand

Power Supply and Distribution: Exascale systems need robust power supply and distribution systems to deliver the high levels of electricity required while maintaining efficiency and reliability.

- **Ex-** High Voltage Direct Current (HDVC) and Uninterruptable Power Supplies and Redundant Power Distribution Systems with atleast a 80 Plus Gold or Titanium Rating from Seasonic or EVGA or Corsair etc

Scalable Storage Solutions: Exascale systems need scalable storage solutions that can handle the massive amount of data generated and processed, including high-speed storage for temporary data and long-term storage for archival purposes.

Networking Infrastructure: High-performance networking infrastructure is essential for connecting exascale systems to external networks and resources, enabling efficient data transfer and communication.

Fault Tolerance and Resilience: Exascale systems require built-in fault tolerance and resilience mechanisms to ensure continued operation in the event of hardware failures or other issues.

6.3 SOFTWARE REQUIREMENTS

Parallel Programming Models: Exascale applications need to be designed and implemented using parallel programming models like MPI (Message Passing Interface) or OpenMP to fully utilize the available computing resources.

- **Ex-** OpenCl, OpenMP

Compilers and Toolchains: Specialized compilers and toolchains are required to optimize code for the specific architecture of exascale systems and achieve high performance.

- **Ex-** LLVM, Intel OneAPI

Operating System: Exascale systems often run on Linux-based operating systems customized for high-performance computing, with optimizations for scalability and performance.

- **Ex-** Red Hat Enterprise Linux (RHEL)

Library Support: Libraries for numerical computations, data analytics, and machine learning are essential for developing exascale applications efficiently.

- **Ex-** CUDA, FFT (Fast Fourier Transform)

Monitoring and Management Tools: Advanced monitoring and management tools are necessary to monitor the performance and health of exascale systems and ensure efficient operation.

- **Ex-** Ganglia , Nagios

Performance Optimization Tools: Tools for profiling, debugging, and optimizing performance are essential for maximizing the efficiency of exascale applications and systems.

- **Ex-** Intel VTune Profiler, AMD CodeXL

Workflow Management Systems: Workflow management systems help organize and manage complex computational workflows in exascale computing environments, ensuring tasks are executed efficiently and reliably.

- **Ex-** Apache Airflow, NextFlow

Security and Privacy Tools: Exascale systems require robust security and privacy tools to protect sensitive data and ensure compliance with regulations and standards.

- **Ex-** OpenSSL, BitLocker, TPM (Trusted Platform Module), Intel SGX

Data Management Software: Data management software is crucial for handling the large volumes of data generated and processed by exascale applications, including data storage, retrieval, and analysis.

- **Ex-** Lustre, Amazon S3, Apache HBase, Collibra

User Interface and Visualization Tools: User interface and visualization tools help users interact with and understand the data and results produced by exascale applications, enabling better decision-making and insights.

- **Ex-** ggplot2, Tableau, ParaView, VTK

CHAPTER 7

FUTURE SCOPE

1. **Advanced Robotics:** Future advancements in AI and robotics will lead to the development of robots that can perform complex tasks in diverse environments. These robots will be equipped with advanced sensors, sophisticated AI algorithms, and the ability to learn and adapt to new situations. They will revolutionize industries such as manufacturing, healthcare, and space exploration by automating tasks that are currently difficult or dangerous for humans.
2. **Autonomous Vehicles:** The future of autonomous vehicles will see the integration of AI algorithms with advanced sensors and communication systems. These vehicles will be able to navigate complex environments, interact with other vehicles and infrastructure, and make split-second decisions to ensure safe and efficient transportation. Self-driving cars and drones will transform the way we travel and transport goods, leading to safer roads and reduced emissions.
3. **Healthcare:** AI will play a pivotal role in the future of healthcare by enabling personalized medicine, improving disease diagnosis and treatment, and accelerating drug discovery. AI algorithms will analyze vast amounts of medical data, including genetic information, patient records, and medical imaging, to provide tailored treatment plans and predictions for individual patients. This will lead to more effective and efficient healthcare delivery, ultimately improving patient outcomes.
4. **Finance:** In the financial sector, AI will continue to drive advancements in financial analysis, fraud detection, risk management, and algorithmic trading. AI-powered systems will analyze market trends, customer behavior, and financial data to make real-time decisions and predictions. This will lead to more informed investment decisions, reduced fraud, and improved risk management strategies.
5. **Natural Language Processing**: Future developments in NLP will enhance our ability to understand and interact with human language. AI-powered systems will be able to comprehend context, sentiment, and nuances in language, leading to more natural and effective communication between humans and machines. This will enable better translation

systems, voice assistants, and chatbots, improving user experiences across various applications.

6. **Climate Change and Sustainability:** AI technologies will play a crucial role in addressing environmental challenges such as climate change and sustainability. AI algorithms will optimize resource management, energy efficiency, and climate modeling, helping to reduce carbon emissions and mitigate the impacts of climate change. AI-powered solutions will also facilitate the transition to renewable energy sources and promote sustainable practices in industries such as agriculture and transportation.
7. **Cybersecurity:** AI will continue to evolve as a key tool in combating cyber threats. AI-powered systems will analyze vast amounts of data to identify and mitigate security risks, detect anomalies, and respond to cyber-attacks in real-time. This will lead to more secure digital environments, protecting critical infrastructure, businesses, and individuals from cyber threats.
8. **Education:** AI has the potential to transform education by personalizing learning experiences, providing adaptive tutoring, and assisting in grading and assessment. AI-powered systems will analyze student performance data to tailor educational content to individual needs, helping students learn at their own pace and style. This will lead to more effective and engaging learning experiences, improving educational outcomes for students worldwide.
9. **Entertainment and Gaming:** AI-driven technologies will enhance entertainment and gaming experiences by creating more immersive and interactive content. AI algorithms will analyze user preferences and behaviors to personalize content, recommend new experiences, and create realistic simulations. This will lead to more engaging and enjoyable entertainment experiences, blurring the lines between reality and virtual worlds.
10. **Ethical and Social Implications:** As AI becomes more integrated into society, addressing ethical and social implications will be paramount. Researchers and policymakers will need to ensure that AI systems are fair, transparent, and accountable. This includes addressing issues such as bias in AI algorithms, data privacy, and the impact of AI on employment and society. Ethical AI development practices will be essential to ensure that AI benefits society as a whole.

CHAPTER 8

CONCLUSION

The convergence of exascale computing and big data analytics represents a critical juncture in the evolution of computing, with profound implications for scientific research, technological innovation, and societal progress. As the demand for computational power continues to grow exponentially, driven by the increasing complexity of scientific simulations, the scale of data generated by modern applications, and the expanding scope of artificial intelligence and machine learning algorithms, the need for advanced computing capabilities has never been more urgent.

Exascale computing, defined by its ability to perform a quintillion (10^18) calculations per second, holds the promise of unlocking new frontiers in scientific discovery and engineering innovation. From simulating complex physical phenomena like climate patterns and astrophysical processes to accelerating drug discovery and materials science, exascale computing has the potential to revolutionize our understanding of the world and our ability to solve pressing global challenges.

However, realizing this potential requires more than just raw computational power. It demands a holistic approach that encompasses not only hardware advancements but also software development, algorithm optimization, and infrastructure scalability. Investment in algorithms, software, and applications is just as critical as in hardware development, as these elements are fundamental to harnessing the full potential of exascale systems.

With a growing emphasis on low-power mobile devices, cloud services, and data analytics. This shift is reshaping the way we approach computing, requiring a collaborative effort between private and global research entities to develop exascale computing and data analytics capabilities that can meet the demands of the future.

In conclusion, the advancement of computing presents both great opportunities and challenges. To realize scientific discoveries through computational science and data analytics, it is essential to maintain a sustained investment in high-performance computing infrastructure. This investment will not only drive technological innovation but also pave the way for groundbreaking discoveries that have the potential to transform our world.

CHAPTER 9

REFERENCES

[1] H. Meuer, H. Simon, E. Strohmaier, et al., (2020, Nov 16). TOP500 super-computer sites [Online]. Available: http://www.top500.org.

[2] G. E. Moore, "Cramming more components onto integrated circuits," Electronics, Vol. 86, No. 1, pp. 82-85, 1965.

[3] R. H. Dennard, F. H. Gaensslen, H. Yu, et al., "Design of ion-implanted MOSFET's with very small physical dimensions," IEEE Journal of Solid-State Circuits, Vol. 9, No. 5, pp. 256-268, 1974.

[4] H. H. Fu, J. F. Liao, J. Z. Yang, et al., " The sunway taihulight supercomputer: system and applications," Science China. Information Sciences, Vol. 59, No. 7, pp. 1-16, 2016.

[5] M. M. Waldrop, "The chips are down for Moore's law," Nature, Vol. 530, pp. 144-147, 2016.

[6] IRDS, (2017). International roadmap for devices and systems [Online]. Available: https://irds.ieee.org/roadmap-2017.

[7] W. Guo, "Research on rtl-level and architecture-level key technique of low power for high performance processor design," Master Thesis, National University of Defense Technology, Changsha, China, 2011. (in Chinese)

[8] D. Wang, S. Shi and H. L. Li, "Three-dimensional technology and low power design: the prospect and challenge of integrated circuits," The 26th Annual Conference of the National Anti-Harsh Environment Computer, Chongqing, China, pp. 24-33, 2016. (in Chinese)

[9] A. Chandrakasan, W. J. Bowhill and F. Fox, Design of HighPerformance Microprocessor Circuits. Wiley-IEEE Press, New York, USA, 2000.

[10] Rabaey, A. Chandrakasan and B. Nilolic, Digital Integrated Circuits: A Design Perspective, 2nd Edition, Pearson Education, Inc., New Jersey, USA, 2003.

ANNEXURE - 1

2021 IEEE 6th International Conference on Intelligent Computing and Signal Processing (ICSP 2021)

Microprocessor Architecture and Design in Exascale Computing Era

WANG Di[1,a], LI Hong-Liang[2,a*]
[a]Jiangnan Institute of Computing Technology
Wuxi, China
*Corresponding author: wzc0425@mail.ustc.edu.cn

2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP) | 978-1-6654-0413-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSP51882.2021.9408861

***Abstract*—In the post exascale computing era, the energy efficiency improvement speed of traditional complementary metal-oxide-semiconductor (CMOS) process has slowed down significantly. In order to realize the zettascale computing capacity in 2035, a great innovation is needed in the microprocessor architecture and design. This paper selects four aspects, which are low power consumption technology, near data processing (NDP) technology, interconnection centered design method and domain specific architecture (DSA), which has a broad development prospect, and focus on the energy efficiency benefits of each technology. Firstly, we analyze various traditional low power consumption technologies and near threshold computing (NTC) technology; secondly, we analyze the NDP technologies such as near memory computing, in memory computing and in network computing; thirdly, we analyze the low overhead network on chip (NOC), NOC supported by new process and cache coherent NOC technology; finally, we take the popular artificial intelligence (AI) processor as an example to analyze the DSA.**

Keywords-post exascale computing; microprocessor; architecture; design

I. INTRODUCTION

High performance computing has become the core support force in national security and national defense construction, economic construction, major projects, basic scientific research and other fields, which is of great significance to national industrial upgrading and structural adjustment. According to the international top 500 high performance computers (TOP500) released in November 2020, Japan's Fugaku high performance computer occupy the top with 442.01Pflop/s (1P=10^{15}) Linpack test performance[1], and there are plans to further upgrade to exascale (10^{18}) in the future. In fact, the HPL-AI test performance of Fugaku has reached 1.42Eflop/s, which has exceeded the exascale. Moreover, the United States, Europe and China have plans to build exascale computers in 2021-2023. It can be said that we are entering the post exascale computing era.

According to the data of the previous TOP500 rankings, the performance of the world's top high performance computers has basically increased by 1000 times every 10 years since the 1990s, exceeding the speed of Moore's Law[2]. In the middle and late years of 2010s, with the failure of Dennard Scaling[3] and the gradual slowing down of Moore's Law, the development speed slowed down significantly, and it is expected to increase by about 100 times every 10 years. Taking Sunway TaihuLight[4] released in 2016 as the symbol of 100Pflop/s, people expect to reach the level of zettascale (10^{21}) around 2035.

The characteristic size of complementary metal-oxide-semiconductor (CMOS) process has reached nanometer level, and the influence of short channel effect, quantum effect, parasitic effect and parameter instability on device performance has become very significant[5]. According to the prediction of the International Roadmap for Devices and Systems (IRDS)[6], there will be about six generations of process progress from 2017 to 2033. If the power consumption of each generation is reduced by 35%, the energy efficiency of the process can be improved by 13 times. However, the performance improvement from 100Pflop/s to 1Zflop/s needs 10000 times. Considering the contribution of process progress, there is still a huge gap of nearly three orders of magnitude, which needs to be filled in the microprocessor architecture and design level.

The purpose of this paper is to look forward to the architecture and design of microprocessor in the era of post exascale computing. In Section 2, various low power consumption technologies are analyzed, including traditional low power consumption technologies and near threshold computing (NTC) technology, In Section 3, various near data processing (NDP) technologies are analyzed, including near memory computing, in memory computing and in network computing. In Section 4, the interconnection centered design method is analyzed, and low overhead network on chip (NOC), NOC supported by new process and cache coherent NOC technology are introduced. In Section 5, we take the popular artificial intelligence (AI) processor as an example to analyze the domain specific architecture (DSA) is analyzed. Finally, summarizes the whole paper.

II. LOW POWER CONSUMPTION TECHNOLOGY

With the increasing integration and working frequency of microprocessors, the requirements of power consumption on power supply and cooling capacity, as well as the stability and reliability problems caused by thermal effect, have increasingly become the bottleneck of the development of high performance computing[7].

According to the generation type, the power consumption of integrated circuit can be divided into dynamic power consumption and static power consumption. Dynamic power consumption is caused by the change of input signal, including the charge and discharge of load capacitor and short-circuit current, which are called flip power consumption and short-circuit power consumption respectively. Static power

978-1-6654-0413-6/21/$31.00 ©2021 IEEE

consumption is caused by leakage current, including leakage current of gate, source and drain and subthreshold current when gate voltage is less than threshold[8].

Flip power consumption

$$P_{sw} = \alpha C_L V_{DD}^2 f \quad (1)$$

where, α is the switching activity, C_L is the load capacitance, V_{DD} is the power supply voltage, and f is the frequency.

Short-circuit power consumption

$$P_{dp} = \alpha t_{sc} V_{DD}^2 I_{peak} f \quad (2)$$

where, t_{sc} is the time for PMOS and NMOS to turn on at the sametime, I_{peak} is the maximum short-circuit current.

Static power consumption

$$P_{stat} = (I_{leak} + I_{sub}) V_{DD} \quad (3)$$

where, I_{leak} is the leakage current and I_{sub} is the subthreshold current. The subthreshold current is closely related to the threshold voltage. The lower the threshold voltage is, the higher the subthreshold current is. In addition, the static power consumption is exponentially related to temperature.

All kinds of low power consumption techniques are based on the analysis of the above power consumption sources.

A. *Traditional Low Power Consumption Technology*

1) *Dynamic Power Consumption Manage Technology*

To reduce the dynamic power consumption, the power supply voltage, load capacitance and switching activity can be reduced.

a) Power Supply Voltage (V_{DD}) Reduction

Reducing the power supply voltage can significantly reduce the dynamic power consumption. However, with the decrease of the power supply voltage, the delay of the circuit increases, which leads to the performance degradation. In order to maintain the performance of the processor, the following methods can be adopted: applying low voltage in non-critical circuits; reducing the threshold voltage to ensure the speed of the circuit; using parallel or pipeline structure to compensate for the reduction of circuit speed[9].

- Multi Voltage Domain: Low voltage is used for the device in the fast path, high voltage is used for the device driving large capacitance, and converter is inserted between low voltage domain and high voltage domain[10].
- Dynamic Voltage and Frequency Scaling: The processor has a very uneven workload when it is running. Keep high power supply voltage and high clock frequency at high load, and reduce power supply voltage and clock frequency at the same time when performing low load work, so as to effectively save energy consumption[11].
- Multi Threshold Transistor: The design is based on high threshold devices, and low threshold devices are used to optimize the timing critical path[12]. This technology is also an important method to reduce static power consumption.
- Structure Optimization: A parallel or pipelined structure can be used to replace the original circuit. Although the latter is larger and more complex, it can achieve the same performance at a lower power supply voltage and get positive benefits[10].

b) Load Capacitance (C_L) Reduction

The input capacitance of CMOS devices is directly proportional to the size of the device. Reducing the size of the device will reduce the speed, which needs to be tradeoff between performance and power consumption.

- Transistor Size Adjustment: Using small size transistor in non-critical path and large size transistor in critical path can effectively reduce the overall power consumption[10].

c) Switching Activity (α) Reduction

Reducing power consumption by reducing unnecessary signal flipping is the most important low power consumption means in architecture design and logic design, including different granularity methods from gate level to system level.

- Transistor Reordering: According to the characteristics of signal activity, the corresponding transistor reordering can greatly reduce their turnover. If the transistor with frequent flip is close to the output end of the circuit, it can prevent the high flip rate of one transistor from spreading to more transistors and bring more power consumption[7].
- Signal Path Equalization: Glitch is the main source of power consumption in complex structures such as arithmetic unit, which may be caused when the length of signal path varies greatly. Choosing the structure with signal path equalization can greatly reduce the probability of glitch[10].
- Operands Isolation: For the functional units without operation, the input is kept at 0 to prevent the turnover of the output signal, so as to reduce the unnecessary dynamic power consumption[7].
- Low Power Consumption Coding: The control logic of the processor is mainly realized by the finite state machine. One of the ways to reduce the power consumption of the control logic is to optimize the state coding mode of the finite state machine, and reduce the signal turnover rate of the circuit by reducing the average distance between two adjacent states[7].
- Clock Gating: Clock power consumption is an important part of processor power consumption. At present, clock power consumption of high performance processor can reach one fourth of the whole chip[13]. Shielding the clock signal in the idle module or reducing it to a very low frequency can save a lot of

power consumption. The flip-flop in the circuit will not flip, but its state value is still saved[12].

- Asynchronous Design: There is no global clock signal in asynchronous design, and the operation of the system is generated by the handshake signal between various components to drive a series of events, which reduces unnecessary flipping in synchronous design[7]. At present, the tradeoff between synchronous circuit and asynchronous circuit, the same frequency and different phase (mesochronous) and globally asynchronous locally synchronous (GALS) are widely used[13].

2) Static Power Consumption Manage Technology

To reduce static power consumption, we can use power off, increase transistor threshold and control chip temperature.

- Power Gating: Power gating is the most effective means to reduce static power consumption. Put the modules with basic synchronization in the working period in a power supply partition. When all the modules in a power supply domain do not need to work, the working voltage of the power supply domain can be turned off to eliminate the static power consumption of the module[8].
- Dynamic Threshold Adjustment: The transistor is used as a four-terminal device, and the threshold of transistor is controlled by substrate bias. For the calculation of low delay, the threshold value is reduced to its minimum value, while for low speed calculation, the threshold value can be increased to minimize leakage current[9].
- Dynamic Thread Assignment and Transfer: Because of the different computing tasks undertaken by each functional unit of the processor, the temperature distribution in space and time is uneven. The operation is transferred from the core with higher temperature to the core with lower temperature, which can effectively control the static power consumption[7].

B. Near Threshold Computing (NTC) Technology

1) Basic Concepts of NTC

Reducing the power supply voltage is the most direct means of low power consumption. Other low power consumption technologies will lead to the aggravation of the "dark silicon" problem, that is, although the chip integrates more transistors, only a small part of the transistors can work at the same time. However, the range of voltage reduction is limited. On the one hand, the decrease of voltage leads to the decrease of conduction current and the increase of circuit delay; on the other hand, the static power consumption in the subthreshold region increases exponentially with the decrease of voltage, and the decrease of voltage will increase the total power consumption.

NTC refers to the circuit where the power supply voltage drops to near the threshold voltage, and its voltage value is between the conventional voltage and the subthreshold voltage[14]. NTC is a tradeoff between performance and power consumption, which can achieve the optimal performance of energy efficiency. When the chip works near the threshold voltage, the energy efficiency can be significantly improved compared with the conventional voltage, as shown in Fig. 1.

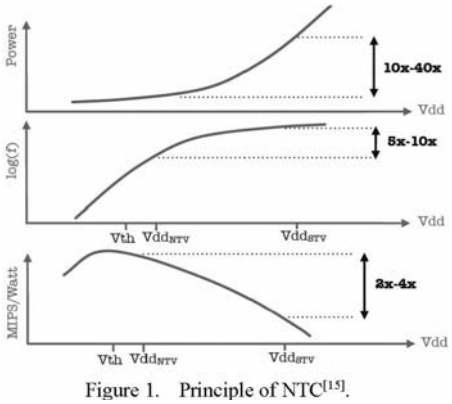


Figure 1. Principle of NTC[15].

Intel released a 32nm process IA-32 architecture processor in 2012[16]. The operating voltage range is 0.28V to 1.2V. When operating at the near threshold voltage of 0.45V, the energy efficiency is 4.7 times higher than that of the standard voltage. In 2015, the KAIST released the target recognition processor with 65nm process[17]. The operating voltage range is 0.5V to 1.2V. When operating at the near threshold voltage of 0.5V, the energy efficiency is 5.8 times higher than that of the standard voltage.

2) Main Challenges of NTC

Although the NTC can achieve 10 times of the standard voltage in power consumption, it will also cause many problems, including the significant decline of circuit performance, the increase of uncertainty of circuit behavior and the significant increase of the risk of circuit functional failure[18]. These problems come from delay deviation which is caused by process deviation, voltage deviation, temperature deviation and aging effect (PVTA). Delay bias has become the biggest challenge of NTC[19].

The process deviation comes from the manufacturing process, which results in the fluctuation of transistor length, width, oxide layer thickness and threshold voltage. The voltage deviation comes from the partial voltage of the interconnect resistance. The more transistors flipped at the same time, the higher the resistance voltage drop. The temperature deviation comes from the change of ambient temperature and the heating of the circuit itself. The aging deviation comes from the decay of transistor life cycle, which leads to the slowdown of transistor speed, the decrease of reliability, the increase of leakage current and the failure of function.

There are a lot of researches on anti-deviation technology in academic circles[19,20]. For example, the anti-fluctuation design technology at the process and device level can enhance the anti-deviation ability of the integrated circuit itself; the static anti-deviation technology can compensate the deviation of the critical path delay by statistical analysis optimization or chip test adjustment, so as to reduce the time series allowance reserved in the design; the dynamic anti-deviation technology based on time series prediction can dynamically predict the time series error through sensor, monitoring circuit, architecture and software information, so that the system can

adjust the voltage and frequency adaptively, so as to reduce the time series margin reserved in the design; the dynamic anti-deviation technology based on timing fault tolerance can detect and correct timing errors in real time, so that the system can still work normally in the case of timing errors, so as to adaptively eliminate the timing margin reserved in the design during operation, and fully improve the energy efficiency of the system.

3) Near Threshold Cache

Cache is the key module of the processor. In recent years, the proportion of area and power consumption is increasing, and the impact on the overall performance of the processor is becoming more and more significant. In the near threshold region, the stability and performance of static random access memory (SRAM) are very sensitive to process deviation, resulting in a sharp increase in the failure probability of SRAM cells[21]. This means that the pursuit of system energy efficiency greatly increases the design difficulty of cache (cache is mainly constructed by SRAM on chip). Research on improving the reliability of near threshold cache focuses on circuit level and architecture level[22].

The research of circuit level mainly focuses on the high reliability design of SRAM circuit at near threshold voltage, which improves the reliability of memory cells at near threshold voltage with large area overhead and delay increase.

The architecture level research mainly includes hybrid cell cache design, error correction, data mapping, data redundancy and cache management strategy redesign. The design of hybrid cell cache is based on the results of circuit level work, which designs cache with high reliability at near threshold voltage. Based on the research of error correction methods, different error correction mechanisms are usually designed by using various error correction code technologies to achieve high cache availability. Data mapping technology maps data to error free areas in the cache. The cache structure based on data redundancy places the redundant data in the low voltage area and the unique data in the high voltage area. Cache management strategy redesign technology can avoid storing data in the wrong cache line by adjusting cache manage strategy.

III. Near Data Processing (NDP)

In the traditional von Neumann architecture, computing, memory and communication are separated. With the development of semiconductor process, the improvement of memory and communication bandwidth is far behind the improvement of computing performance. Since 1990s, the "memory wall" and "communication wall" have become more and more key factors to restrict the performance improvement of processor. The more serious problem is that the energy consumption of data movement has even exceeded the calculation energy consumption. Taking Intel research on 7nm process processor as an example[23], the memory access power consumption reaches 45.5%, and the communication power consumption reaches 18.2%. This means that data movement is becoming a bottleneck for the improvement of processor performance and power efficiency.

Increasing cache level and capacity can reduce the data movement quantity to a certain extent, but it is far from fundamentally solving this problem. Therefore, the idea of NDP that closely integrated with data and processing, which has been widely concerned by academia and industry.

A. Near Memory Computing

The basic idea of near memory computing is to close the data to the computing unit, so as to reduce the delay and power consumption of data movement. In near memory computing, logic or processing units are closer to memory. However, memory and computing units are still separate parts[24].

1) Near Memory Computing Based on SRAM

Near memory computing based on SRAM is mainly a multi-level memory architecture. A series of cache or local memory are inserted between the computing unit and the main memory. The temporal and spatial locality of the program are used to reduce the data movement distance.

2) Near Memory Computing Based on DRAM

In the 1990s, in order to break through the limitation of memory wall, a large number of research on near memory computing technology based on dynamic random access memory (DRAM) appeared. For example, the IRAM proposed by Patterson et al.[25] is manufactured by standard DRAM process, and vector processor is integrated into memory chip to greatly reduce access delay of processor to memory and make full use of memory bandwidth; the FlexRAM proposed by Kang et al.[26] adopts a tightly coupled architecture, and the computing array composed of 64 reduced instruction set computing (RISC) processor cores and DRAM cells are interleaved, which can make deep use of DRAM memory bandwidth and obtain significant performance improvement in data mining, decision system and other applications. Due to the incompatibility between the DRAM process and the logic process of the processor core, and the problem of DRAM access is alleviated to a certain extent by increasing and optimized cache design, the related research has not been well applied[27].

In recent years, with the emergence of advanced packaging technology and three dimensional integration, DRAM based near memory computing technology has gained a new development opportunity. Hybrid memory cube (HMC) and high bandwidth memory (HBM) are two new types of stacked DRAM based on through silicon via technology.

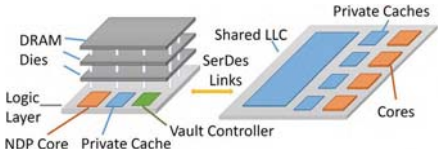


Figure 2. Near memory computing based on HMC[28].

HMC and HBM use stack to improve memory density and memory capacity, and improve memory bandwidth by high speed serial transmission or parallel width. They have the advantages of high integration, high bandwidth and high energy efficiency. Moreover, with the aid of stacking technology, the logical layer and memory layer of different manufacturing processes can be stacked together, and vertical

multiple memory layers correspond to one logical layer. Encapsulating the logical layer and memory layer reduces data access latency and power consumption[29].

B. *In Memory Computing*

There is still data movement from memory to computing unit in near memory computing. In order to eliminate the cost of data movement, a large number of in memory computing research has appeared in the academic circles. In memory computing is to perform computation in memory array, which realizes the complete integration of memory and computing.

1) *In Memory Computing Based on SRAM*

Jeloka et al.[30] divides the word line of 6T SRAM into left and right lines. At the same time, the differential sensitive amplifier is transformed into two single end cross coupled sensitive amplifiers. Through the control of two word lines and two sensitive amplifiers and the addition of logic gates at the output end of the sensitive amplifier, the basic logic of and, or, and non is realized. On this basis, Aga et al.[31] implements a compute cache that supports no carry multiplication. Eckert et al.[32] further implements addition, multiplication and subtraction operations, and proposed neural cache for deep learning algorithm.

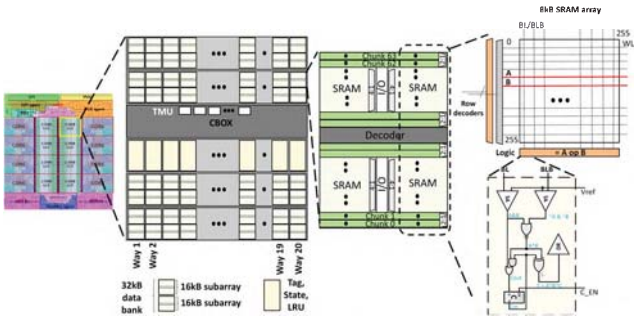


Figure 3. Neural cache architecture[32].

Kang et al.[33] vertically stored *n*-bit binary numbers by bit, so that different bits share the same bit line. Through the control signal, all memory bits are read at the same time, and the word line gating time from high bit to low bit of the data is binary weighted, so that there is a digital to analog conversion relationship between the voltage drop on the bit line and the stored number. Then the analog processing circuit module is used to process the voltage drop signal, so as to quickly calculate |A−B| and A•B (A and B are two vectors), and then the calculated value is converted into digital signal by analog-to-digital converter (ADC). By accelerating the calculation of vector distance and dot product in SRAM, the operation efficiency of AI algorithm can be greatly improved.

2) *In Memory Computing Based on DRAM*

The DRISA architecture proposed by Li et al.[34] implements convolutional neural network (CNN) computation based on DRAM process, providing large scale on chip memory and high computational performance. By redesigning the bit line of DRAM array, DRISA realizes simple logic and shift circuits, and uses these circuits to support complex operations such as addition and multiplication. Compared with GPU, DRISA can improve the energy efficiency of integer operation by 15 times.

3) *In Memory Computing Based on NVM*[29]

In recent years, nonvolatile memories (NVMs) such as flash, STTRAM, PCM and RRAM have developed rapidly. Due to the natural fusion of computing and memory, NVM is very suitable for in memory computing[35]. All kinds of NVMs are gradually moving towards the practical stage, and it is possible to apply them in microprocessor in the future.

In 2016, IBM created the first artificial nano scale random phase change neuron, which can be used to create artificial neurons. The membrane potential of the artificial neuron can be expressed by the phase structure of the nano phase change device. In 2018, IBM proposed to accelerate the training of fully connected neural network by PCM to perform calculation in the data storage location. The energy efficiency of the chip is 280 times that of GPU, and it can achieve 100 times of computing performance in the same area.

In 2010, HP Labs annouced that RRAM has Boolean logic operation function, which means that computing and memory functions can be integrated in RRAM. The first example of using RRAM to realize logical storage fusion is IMP proposed by Borghetti et al. In 2018, the PRIME architecture proposed by Xie et al. implemented neural network computing based on RRAM. The power consumption of PRIME can be reduced by 20 times and the speed can be increased by 50 times when it is fabricated in 15nm process.

C. *In Network Computing*

In network computing is a frontier topic in the field of high performance computing and AI. It effectively solves the problems of collective communication and point-to-point bottleneck in application, and provides a new idea and scheme for the scalability of data center. In network computing use network cards, switches and other network devices to calculate data online during data transmission, so as to reduce communication delay and improve overall computing efficiency[36]. The idea of in network computing can also be applied to the NOC with a single processor.

Zheng et al.[37] proposed a multi-mode data-flow transmission for many core processors. The data-flow transmission is asynchronous with the core pipeline, which makes it easy to prefetch data and effectively supports memory access delay hiding. The function of data-flow transmission is mainly completed by the stream transmission engine, which supports the concurrent processing of multiple transmissions. One of the main characteristics of data stream transmission is to support multiple stream transmission modes according to the structure characteristics and application requirements. These transmission modes can make the data distributed in a multi-dimensional way, effectively improve the efficiency of data localization and save memory access bandwidth.

Huang et al.[38] studied the idea of mapping the computing kernel to the memory network, so as to play a role in the data-flow mode of in network computing through NDP. They proposed an in network computing architecture called Active-Routing. By using the aggregation mode of arithmetic

operation intermediate results, the computing can be carried out in the process of approaching data processing. The architecture utilizes large-scale memory level parallelism and network concurrency to optimize aggregation operations along a dynamically constructed active routing tree. Compared with the advanced memory processing architecture, Active-Routing reduces the energy delay product by 80%.

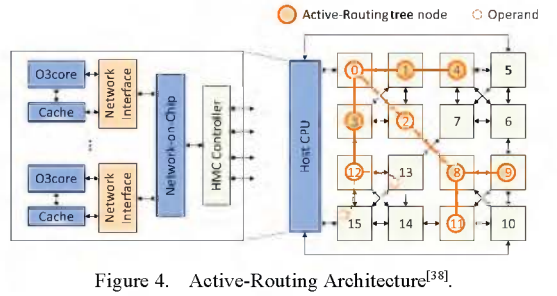


Figure 4. Active-Routing Architecture[38].

Compression processing in the process of data transmission is an important means to improve memory capacity and bandwidth and reduce data transmission power consumption. IBM has added a memory compression acceleration module NXU[39] to Z15 processor, which only increases the area of the processor by less than 0.5%, improves the compression efficiency by 388 times, and achieves a compression bandwidth of 280GB/s. NVIDIA also proposes a data compression technology called Buddy Compression for GPU to improve the effective capacity of on-chip memory and application performance.

IV. Interconnection Centered Design Method

At present, the development of processor has entered the stage of many core processor integrating dozens or even hundreds of cores. The design of many core processor needs to adopt the "interconnection centered" design method[41]. On the one hand, in the case of abundant computing resources, the efficiency of interconnection layer largely determines the performance of many core processors. On the other hand, the interconnection layer brings a lot of power consumption, so the design of low power many core processor must reduce the power consumption of the interconnection layer.

The improvement of processor integration means that the width of interconnects inside the chip becomes smaller and smaller, and the delay of signal transmission per unit distance increases accordingly. In the early CMOS circuits, the influence of the connection on the circuit performance and power consumption is ignored. The connection transmits signals at an almost infinite speed without power consumption and coupling effect. With the development of process technology, the influence of wire connection gradually appears. The parasitic effects such as capacitance, resistance and inductor affect the performance, power consumption and reliability of the system. In particular, the impact of global connection on delay and power consumption increases with the reduction of process scale [13].

At the beginning of the 21st century, NOC was proposed as a new design paradigm of interconnection communication on chip[42,43,44]. NOC replaces the traditional bus architecture with the packet communication architecture of point-to-point communication, which can reduce the chip area and power consumption, and improve the performance and scalability of the system.

A. Low Overhead NOC

In general, NOC uses routing nodes to connect processor cores into an interconnection network, and message exchange is used to communicate between cores, so as to form the on-chip communication system of processor. In the past 20 years, researchers have done a lot of research on topology, routing algorithm, router structure, switching technology, flow control technology, virtual channel technology, buffer implementation, error correction and coding, transmission link, network interface, QoS, program mapping, etc. Among them, bufferless router and router free network on chip are two important theoretical developments.

1) Bufferless Router[45]

Before bufferless router was put forward, network on chip (NOC) used wormhole or virtual channel router more often. The characteristic of NOC is that every input or output port of router contains buffered packets. Although the buffer can effectively improve the bandwidth utilization of the network, reduce packet loss and bypass routing, it also has the problems of consuming a lot of energy, complex flow control strategy and occupying a large area. Therefore, bufferless router emerges as the times require, which provides a low overhead solution for network on chip.

In the bufferless router, there is no extra buffer in the router except pipeline register, which can greatly reduce the energy consumption and area overhead of the router, and simplify the design of the router.

Bufferless routers can be divided into drop based routers and deflection based routers. In a packet loss based bufferless router, if a header microchip arrives at the router and the required output port is busy, all the microchips of the packet are discarded. In the bufferless router based on deflection routing, the router immediately forwards the packet to the next router after receiving it. In the case of competition, the packet can deviate from the shortest path routing.

2) Routerless NOC

Ring topology has long been considered as poor scalability. However, the isolated multi ring (IMR) architecture proposed by Liu et al.[46] can even support 1024 core processors. In IMR, any pair of cores are connected through at least one isolation ring, so that each packet can reach the destination without being transmitted from one ring to another. Therefore, IMR no longer needs expensive routers to build the grid network, which not only improves the network performance, but also reduces the hardware overhead. The experimental results show that IMR has significant advantages in bandwidth and delay, while reducing area and power consumption.

On the basis of IMR, Alazemi et al.[47] proposed the concept of routerless NOC. Routerless NOC completely eliminates the router, cleverly uses the routing resources, and achieves the same hop count and scalability as router based NOC. The evaluation results show that compared with the traditional grid,

the power consumption, area, zero load packet delay and bandwidth of the routerless NOC are reduced by 9.5 times, 7.2 times, 2.5 times and 1.7 times respectively.

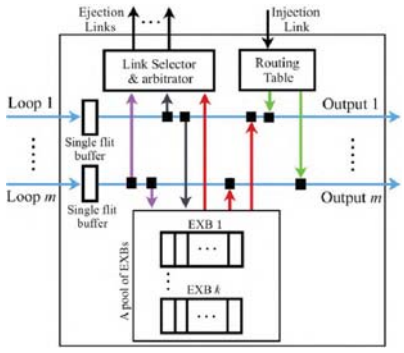


Figure 5. Interface module of routerless NOC[47].

From the development status, EDA tool support and physical design friendliness are still problems to be solved in routerless NOC.

B. *The Combination of NOC and New Process*

New processes bring new opportunities for the development of network on chip. The development of three dimensional integration technology and the progress of optical interconnection technology on chip will bring great changes to the network on chip architecture.

1) *Three Dimensional NOC*

Three dimensional integration is a technology to realize vertical interconnection between through silicon vias, which has the advantages of reducing the global interconnect length and increasing the interconnect density. Limited by the traditional two-dimensional architecture, NOC still cannot fundamentally avoid a series of related problems, such as global connection too long, connection delay, power consumption and so on. 3D NOC technology, which combines NOC and 3D integration technology, realizes inter core interconnection with 3D architecture to obtain better performance[48].

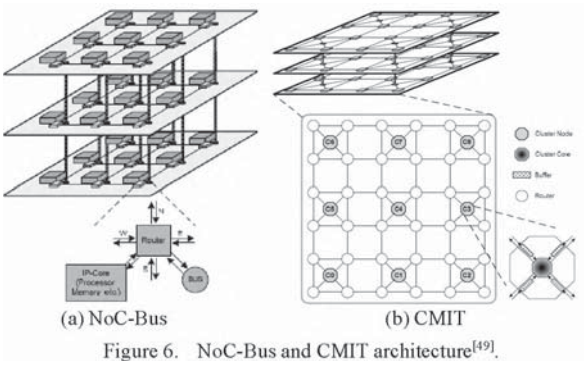


Figure 6. NoC-Bus and CMIT architecture[49].

Li et al.[50] proposed a structure called NOC-Bus for TSV delay and power consumption which are far less than the global interconnects in silicon chips. The structure uses mesh structure in each silicon chip, and uses bus to interconnect between silicon chips. Compared with 3D mesh, this structure can reduce the area and power consumption, and reduce the zero load delay. On the basis of NOC bus, Masoud et al.[49] proposed a 3D NOC structure called CMIT. The interconnection on each silicon chip also uses mesh network. Four nodes on each silicon chip are connected to a collection node, and the collection nodes on different silicon chips are connected by bus.

Park et al.[51] proposed a three-dimensional router architecture MIRA, which distributes the data channels between routers on different silicon wafers, and can reduce the area requirement and power consumption of 3D NOC. Kim et al.[52] proposed a dimension decomposition 3D NOC router 3D DIMDE, which decomposes the crossbar in the router into three modules, and the scale of each module is 2×2. Every time a message crosses a dimension, it will increase the delay of a clock cycle, which can achieve better performance under the dimension routing algorithm. Feng et al.[53] proposed a single cycle high-performance bufferless router for 3D NOC, which uses three segment permutation network instead of continuous switch distributor and 7×7 cross switch.

2) *Optical NOC*

In the field of high performance computer, optical interconnection has already shown its advantages in the network interconnection among cabinets, printed circuit boards and even chips. The progress of CMOS compatible nanophotonics technology provides the conditions for the development of optical network on chip. The bit rate transparency of optical media makes high-speed and low-power data transmission possible. The low loss characteristic of signal propagation in optical waveguide can increase the distance bandwidth product of the network, making the data can be transmitted further[11].

Some researches use passive wavelength switching to realize optical network on chip. Briere et al.[54] proposed a multilevel non-blocking optical router: λ-router. λ-router uses a passive switching structure based on wavelength, and uses wavelength division multiplexing (WDM) technology to realize non-blocking switching. Then, based on λ-router, an optical NOC using different wavelengths to realize optical switching is proposed. Batten et al.[55] proposed an optical interconnection structure based on silicon-based nano optical communication technology and using local mesh/global switching (LMGS) method, which connects the on-chip processor cores interconnected by grid structure to the off chip global switching structure.

There are also some researches on the implementation of on-chip optical networks using active optical switching units. Shacham et al.[56] proposed an optoelectronic hybrid network on chip, in which the optical interconnection network is used for high bandwidth message transmission, and the electrical network with the same topology is used to control the optical network and send short messages. Before sending the message through the optical network, the short control message is transmitted to the destination node through the electrical network, so as to reserve the optical device resources on the path. Mo et al.[57] proposed a hierarchical hybrid optoelectronic NOC architecture home, which uses optoelectronic hybrid router to realize wormhole switching in local network. At the same time, circuit switching is used to serve the global network.

C. Cache Coherent NOC

Cache coherence protocol is a mechanism to propagate the newly written value of one core to other cores to ensure that all cores see the coherent shared storage content. Considering that cache coherent programming mode can reduce the burden of programmers compared with message passing programming mode, cache coherence protocol will continue to exist in order to be compatible with a large number of historical code based on cache coherent programming mode. Therefore, it is necessary to provide effective communication support for cache coherence protocol[58].

Cheng et al.[59] observed that different cache coherence messages have different sensitivity to delay and bandwidth in the collaborative design of NOC and cache coherence protocol. Therefore, heterogeneous connection is proposed to transmit different types of messages. Delay sensitive messages are transmitted through low delay connection, and bandwidth sensitive messages are transmitted through high bandwidth connection. Eisley et al.[60] proposed to store the directory information of the directory cache coherence protocol in the NOC router, so as to reduce the latency of cache read-write transactions. Agarwal et al.[61] implemented message ordering at the NOC layer to support the implementation of broadcast cache coherence protocol on unordered network. On this basis, Agarwal et al.[62] further proposed to set a filter on the router to eliminate some unnecessary listening messages. Based on a similar idea, Jerger[63] uses a filter to eliminate redundant cache line void messages in coarse-grained directory cache coherence protocol.

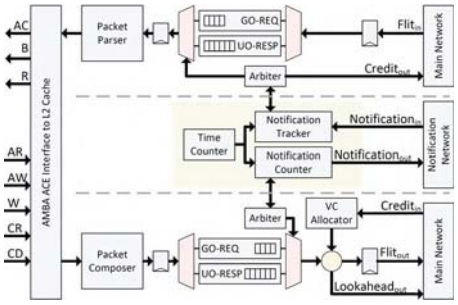


Figure 7. Network interface controller of SCORPIO architecture[64].

SCORPIO proposed by Daya et al.[64] is an architecture using broadcast cache coherence protocol, which has an independent fixed delay, bufferless mesh structure NOC and can realize distributed global sorting. Message delivery is separated from sorting, allowing messages to arrive at any time and in any order, while still maintaining the correct order. The main network is an unordered network and is responsible for broadcasting the actual coherence request to all other nodes. The notification network is used to broadcast the notification message of each coherence request sent in the primary network to all nodes. The notification message uses bit vector to represent the request source, so broadcasting can combine bit vectors by bit-or operation without competition.

Hu et al.[65] proposed a cooperative design of heterogeneous interconnection communication and cache coherence based on transmission line. Using transmission line to build NOC, it can provide low delay and low power consumption NOC; combining transmission line NOC with traditional mesh network, it can build on-chip heterogeneous interconnection system. It optimizes the adaptability of cache coherence protocol and reduces the maintenance cost of directory based cache consistency. Through the hardware real-time monitoring of the time locality of data, the system dynamically adjusts the storage strategy of shared read-write data to reduce the maintenance of cache coherence. According to the time delay sensitivity, messages with different characteristics can dynamically select the appropriate interconnection network transmission, so as to reduce the on-chip delay and improve the adaptability of cache coherence.

V. Domain Specific Architecture (DSA)

In the past 20 years, the technology of instruction level parallelism has not made great progress, and the improvement of processor performance mainly depends on the increase of the number of cores. However, the performance improvement efficiency of the processor is limited by the parallelism of the application itself. In addition, the increase of the number of cores cannot significantly improve the power efficiency of the system.

Processor has the advantages of high flexibility and programmability, but also has the problem of low power efficiency. Application specific integrated circuit (ASIC) gives up the programmable ability, but it can perform thousands of operations in parallel for specific applications, which greatly improves the performance and power efficiency. Compared with ASIC, the power efficiency of processor can be tens of times or even hundreds of times.

Pure processor has essential and insurmountable obstacles in performance and power efficiency, while pure accelerator for specific fields has great limitations in flexibility and programmability. Hennessy and Patterson, winners of Turing prize, have repeatedly emphasized that domain specific processors will be the main trend in the future[66,67]. In the future, the most important mode of chip system will be to integrate general multi-core processor and special accelerator to form a "general core+accelerator" system, so as to obtain the programmability and flexibility of general processor as well as the performance and power efficiency of application specific accelerator[68].

Taking the popular AI processor as an example, this paper introduces the architecture, design principle and implementation method of DSA. Relevant contents mainly come from [69], [70] and [71].

A. Case Study of DSA

In 2010, Temam[72] elaborated the significant influence that neural network may bring to hardware design in various fields of general computing and special computing. Since then, the design of AI chips has entered a period of rapid development, with the emergence of many DSA processors in the field of AI represented by Google's tensor processing unit (TPU)[69] and Huawei's Ascend[70].

1) TPU Architecture

The main modules of TPU include systolic array, vector computing unit, main interface module, queue module, unified buffer and DMA control module.

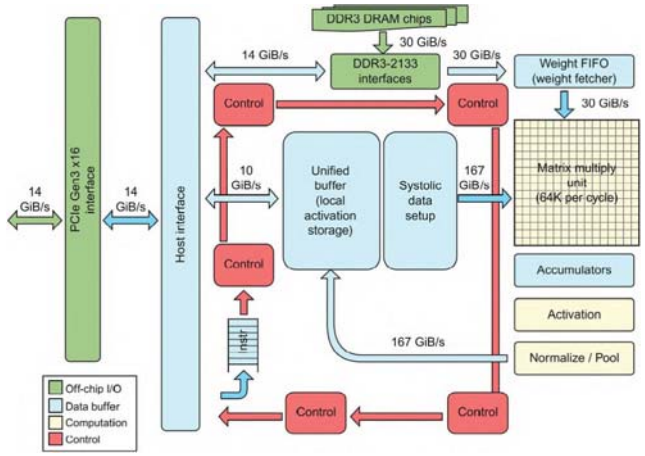


Figure 8. TPU architecture[69].

The main interface is used to obtain the parameters and configuration of neural network, such as the number of network layers, multi-layer weight and activation value. After receiving the read command, DMA control module will read and store the input feature and weight data in the unified on-chip buffer. At the same time, the main interface sends the execution instruction to the queue module. After receiving the instruction, the queue module starts and controls the calculation mode of the whole neural network, such as how the weights and eigenvalues enter into the pulsating array and how to accumulate them in blocks. The main function of the unified buffer is to store the intermediate results of input and output, and also to send the intermediate results to the systolic array again for the next layer calculation. The queue module can send control signals to unified buffer, pulse array and vector computing unit, and can also communicate directly with DMA control module and memory.

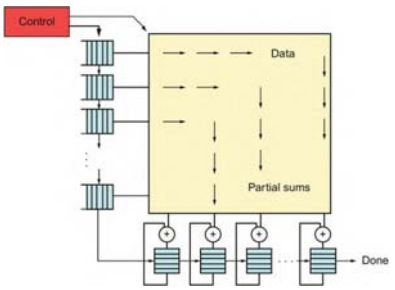


Figure 9. Systolic structure of TPU[69].

Systolic array is used to accelerate convolution. The main body of systolic array is a two-dimensional sliding array, in which each node is a systolic computing unit, which can complete a multiplication and addition operation in a clock cycle, and realize the right and down sliding transmission of data between the computing units of each row and column through horizontal or vertical data path.

2) Ascend Architecture

The main components of Ascend include control CPU, AI computing engine (including AI core and AI CPU), multi-layer cache or buffer, digital vision pre-processing (DVPP), etc.

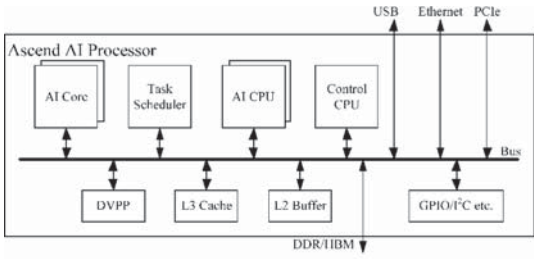


Figure 10. Ascend AI processor architecture[70].

Ascend AI processor integrates multiple CPU cores, each core has its own L1 and L2 cache, and all cores share an on-chip L3 cache. According to the function, the integrated CPU core can be divided into control CPU dedicated to control the overall operation of the processor and AI CPU dedicated to undertake non matrix complex computing. In addition to CPU, the real computing power of the processor is the AI core based on Da Vinci architecture.

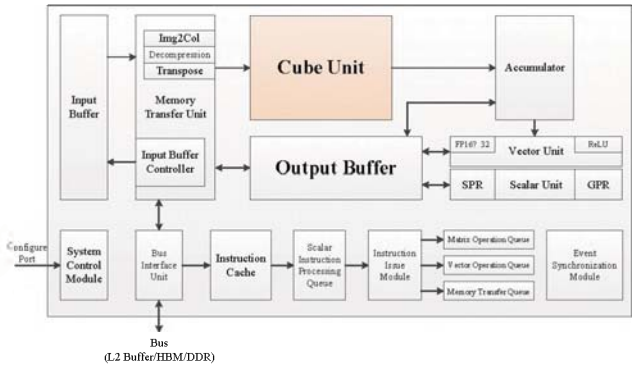


Figure 11. Da Vinci architecture[70].

Da Vinci architecture includes three basic computing resources: cube unit, vector unit and scalar unit. These three computing units correspond to three common computing modes: tensor, vector and scalar, forming three independent execution pipelines.

In order to coordinate the data transmission and transportation in AI core, a series of on-chip buffers are distributed around the three computing resources. Input buffer (IB) and output buffer (OB) are used to place the whole image feature data, network parameters and intermediate results. After the input buffer, a memory transfer unit (MTE) is set to realize data format conversion functions such as transpose with high efficiency. In addition, there are some high-speed register units used to provide temporary variables in each calculation unit.

B. Design Principle of DSA

Hennessy and Patterson systematically summarized the design principles of DSA, including the following five principles[71].

1) Using special memory to minimize the distance of data movement.

The multi-level cache in general-purpose microprocessors uses a lot of area and energy to optimize the data movement of programs. DSA compiler writers and programmers know their domain, so they don't need hardware to move data for them. Instead, software controlled memory is dedicated to specific functions within the domain and tailored to reduce data movement.

- TPU has a 24MB unified buffer, which stores the intermediate matrix and vector of MLP and LSTM, as well as the feature map of CNN. It is optimized for each 256B access. It also has 4MB accumulators, each 32-bit wide, which collect the output of the matrix cells and act as the input of the hardware for calculating the nonlinear functions. The 8-bit weights are stored in a separate off chip weight memory dram and accessed through the on-chip weight FIFO.
- In view of the characteristics of deep neural network, such as large number of parameters and many intermediate values, Ascend has equipped an 8MB on-chip buffer (L2 buffer) for AI computing engine to provide high bandwidth, low latency and efficient data exchange and access.

2) The resources saved from abandoning advanced microarchitecture optimization are put into more computing units or larger memory.

With out-of-order execution, multi-thread, multi-core, prefetch and address coalescing, architects translate the benefits of Moore's Law into resource intensive optimization of CPU and GPU. Considering the deeper understanding of program execution in these narrow areas, it is better to spend these resources on more processing units or larger on-chip storage.

- TPU provides 28MB of dedicated storage and 65536 8-bit ALUs, which means it has about 60% of the storage and 250 times the ALU of the server level CPU, although its size and power consumption are only half of the server level CPU. Compared with the server level GPU, the storage on chip of TPU is 3.5 times that of GPU, and Alu is 25 times that of GPU.
- 256 matrix calculation sub circuits are integrated in the matrix calculation unit of Acsend AI core, and each sub circuit realizes two 16 element vector dot products (each element is a 16 bit floating-point number). Two 16×16 matrices can be multiplied by one instruction, which is equivalent to 16^3=4096 multiplication and addition operations in a very short time.

3) Use the simplest parallel form that matches the domain.

The target domain of DSA almost always has inherent parallelism. The key decision of DSA is how to use this parallelism and how to open it to software. It is necessary to design DSA around the natural granularity of parallelism and simply expose the parallelism in the programming model.

- The performance of TPU is provided by a two-dimensional SIMD parallel processing unit. Its 256×256 matrix multiplication unit adopts pulsating organization and a simple instruction overlapping pipeline.
- In view of the fact that the demand of large computing power in AI applications can often be transformed into matrix operation, Ascend provides its performance through a fixed 16×16 matrix operation unit. This is similar to NVIDIA's Tensor Core[73].

4) Reduce data size and type to the simplest size and type required by the domain.

Applications in many fields are usually limited in storage. Therefore, by using narrower data types, effective storage bandwidth and on-chip storage utilization can be improved. Narrower and simpler data also allows designers to place more computing units in the same chip area.

- TPU mainly computes 8-bit integers, although it supports 16 bit integers and accumulates them in 32-bit integers.
- The matrix computing unit in Ascend AI core supports 8-bit integer and 16 bit floating-point computation, while the vector computing unit supports 16 bit and 32-bit floating-point computation as well as a variety of integer computation.

5) Using domain specific programming language to port code to DSA.

A typical challenge for DSA is to make applications run on a new architecture. In fact, domain specific programming languages have long been popular, such as halide for video processing and tensorflow for deep learning. Such a language makes it more feasible to port applications to DSA. In some areas, only a small number of applications need to run on DSA, which also simplifies the migration.

- TPU is programmed with TensorFlow programming framework. TensorFlow supports running on CPU, GPU and TPU, and its programming style is declarative programming.
- Acsend adopts MindSpore programming framework. MindSpore uses functional differentiable programming architecture to achieve dynamic static combination of development and debugging mode, automatically complete model segmentation and tuning, and provide consistent development, on-demand collaboration and flexible deployment functions.

In the field of AI, major companies develop programming frameworks for their own hardware. The work of various programming frameworks is basically similar. By defining a set of intermediate representations dedicated to deep learning, we can get through a process of DSL→Deep Learning IR→LLVM IR→Target, and add various optimizations in the middle.

C. Implementation Method of DSA

1) Implementation Method Based on IP Block

Amdahl's Law reminds us that the performance of the accelerator is limited by the rate of data transmission between the core and the accelerator. Integrating the core and accelerator into the same SOC will benefit the application.

This design is called IP block, which is usually specified by hardware description language to integrate into SOC. Many companies produce IP blocks, and other companies can buy these IP blocks to build SOC for their own applications without having to design everything themselves.

IP block must be scalable in area, power consumption and performance. For a new IP block, it is particularly important to provide a small resource version, because it may not have a good foothold in the SOC ecosystem. If resource requests are moderate, adoption is much easier.

2) Open Source, Extensible Instruction Set Architecture

For DSA designers, an open source and extensible instruction set architecture is needed.

On the one hand, a challenge for DSA designers is to determine how to work with the CPU to run the rest of the application, which means choosing the instruction set architecture of the CPU.

On the other hand, in order to cover as many applications as possible, universal instruction set architecture often needs to support thousands of instructions, which leads to the complexity of pipeline front-end design (finger fetching, decoding, branch prediction, etc.), which has a negative impact on performance and power consumption. Domain specific instructions can greatly reduce the number of instructions, increase operation granularity, integrate memory access optimization, and achieve energy efficiency improvement[74].

RISC-V is a free and open instruction set architecture, which reserves a lot of opcode space for domain specific coprocessor to add instructions, so as to achieve closer integration between core and accelerator. Due to the openness of RISC-V, designers can obtain the IP block of RISC-V for free and get the support of open source software.

3) Microarchitecture Design[27]

The microarchitecture design of DSA processor is a process of software and hardware co design. The design steps include: ①design and analyze the target application algorithm, locate the hot area of operation and data access storage in the target application algorithm, and identify the bottleneck of acceleration; ②transfer the software code of calculation time from the benchmark processor to the special functional unit module, and at the same time, update the data in the benchmark processor On the basis of processor, add pipeline, special register, local register, special operation unit and corresponding acceleration instructions to form a new hardware model; ③ design software development tools, mainly including instruction set simulator, software compiler, assembler and linker; ④simulate target application and evaluate acceleration performance based on new hardware model and software development tools; ⑤ according to the evaluation results, iterative optimization is carried out repeatedly to meet the preset target requirements; ⑥DSA processor is implemented based on FPGA or ASIC to further evaluate the speed, area and power consumption.

4) Software Stack

In order to make DSA processor play an excellent performance, it is very important to design a perfect software solution. A complete software stack includes a framework for computing resources and performance tuning, as well as various supporting tools.

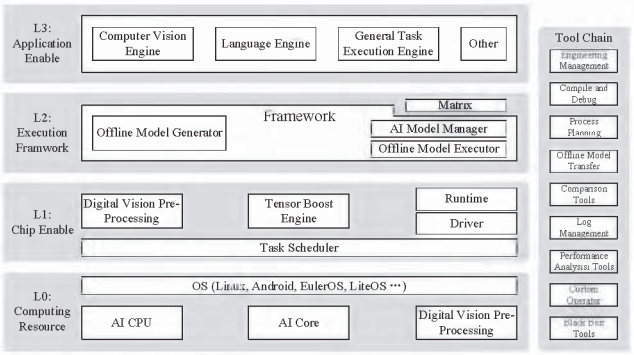


Figure 12. Software stack of Ascend AI processor[70].

Fig. 12 shows the software stack of Acsend AI processor, which is divided into two parts: neural network software flow and tool chain. Neural network software flow mainly includes matrix, framework, runtime, DVPP, tensor boost engine (TBE) and so on. Neural network software flow is mainly used to complete the generation, loading and execution of neural network model. The tool chain includes engineering management, compilation and debugging, process planning, offline model conversion, comparison tools, log management, performance analysis tools, custom operators and black box tools. Tool chain mainly provides auxiliary convenience for the realization of neural network.

VI. CONCLUSION

In the post exascale computing era, the development of CMOS process is difficult to maintain the original speed of energy efficiency progress. In order to achieve the goal of zettascale computing around 2035, we need to fill the energy efficiency gap of nearly three orders of magnitude in processor architecture and design. The full application of traditional low-power consumption technology and the development of NTC technology are expected to improve the energy efficiency by about one order of magnitude. NDP technology minimizes data mobility, while the interconnection centered design method minimizes the overhead of data mobility. The combination of the two technologies is expected to further improve the energy efficiency by about one order of magnitude. In recent years, DSA processor has developed rapidly. From the existing results, it has been proved that it can improve the energy efficiency by more than one order of magnitude. In summary, architecture and design innovation can support the sustainable development of processor technology in the post exascale era.

The development of technology will not stop, and the human pursuit of computing peak will continue. It can be predicted that in the post exascale computing era, the evolution of microprocessor architecture and design will be more exciting.

REFERENCES

[1] H. Meuer, H. Simon, E. Strohmaier, et al., (2020, Nov 16). *TOP500 super-computer sites* [Online]. Available: http://www.top500.org.

[2] G. E. Moore, "Cramming more components onto integrated circuits," Electronics, Vol. 86, No. 1, pp. 82-85, 1965.

[3] R. H. Dennard, F. H. Gaensslen, H. Yu, et al., "Design of ion-implanted MOSFET's with very small physical dimensions," IEEE Journal of Solid-State Circuits, Vol. 9, No. 5, pp. 256-268, 1974.

[4] H. H. Fu, J. F. Liao, J. Z. Yang, et al., "The sunway taihulight supercomputer: system and applications," Science China. Information Sciences, Vol. 59, No. 7, pp. 1-16, 2016.

[5] M. M. Waldrop, "The chips are down for Moore's law," Nature, Vol. 530, pp. 144-147, 2016.

[6] IRDS, (2017). *International roadmap for devices and systems* [Online]. Available: https://irds.ieee.org/roadmap-2017.

[7] W. Guo, "Research on rtl-level and architecture-level key technique of low power for high performance processor design," Master Thesis, National University of Defense Technology, Changsha, China, 2011. (in Chinese)

[8] D. Wang, S. Shi and H. L. Li, "Three-dimensional technology and low power design: the prospect and challenge of integrated circuits," The 26th Annual Conference of the National Anti-Harsh Environment Computer, Chongqing, China, pp. 24-33, 2016. (in Chinese)

[9] A. Chandrakasan, W. J. Bowhill and F. Fox, Design of High-Performance Microprocessor Circuits. Wiley-IEEE Press, New York, USA, 2000.

[10] Rabaey, A. Chandrakasan and B. Nilolić, Digital Integrated Circuits: A Design Perspective, 2nd Edition, Pearson Education, Inc., New Jersey, USA, 2003.

[11] J. H. Wang, "Low-power on-chip networks in high-performance multi-core processors," Doctor Thesis, National University of Defense Technology, Changsha, China, 2014. (in Chinese)

[12] D. Wang and M. J. Wang, "High performance many-core processor: a design method based on ratio of performance and power estimation," High Performance Computing Technology, No. 233, pp. 27-33, 2015. (in Chinese)

[13] Z. Y. Yu, X. Y. Zeng and S. J. Wei, Microprocessor Design: Architecture, Circuit and Implementation. Science Press, Beijing, China, 2019. (in Chinese)

[14] I. C. Wey, P. J. Lin, B. C. Wu, et al., "Near-threshold-voltage circuit design: The design challenges and chances," International SoC Design Conference (ISOCC), Jeju, Korea (South), pp. 138-141, 2014.

[15] R. G. Dreslinski, M. Wieckowski, D. Blaauw, et al. "Near-threshold computing: reclaiming moore's law through energy efficient integrated circuits," Proceedings of the IEEE, Vol. 98, No. 2, pp. 253-266, 2010.

[16] S. Jain, S. Khare, S. Yada, et al. "A 280mv-to-1.2v wide-operating-range ia-32 processor in 32nm cmos," IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, USA, pp. 19-23, 2012.

[17] H. Yoo, Y. Kim and I. Hong, "A 0.5V 54μW ultra-low-power recognition processor with 93.5% accuracy geometric vocabulary tree and 47.5% database compression," IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, USA, pp.330-331, 2015.

[18] K. Yang, "Low power sram research and design under near-threshold voltage supply," Master Thesis, Shanghai Jiao Tong University, Shanghai, China, 2011. (in Chinese)

[19] S. Wang, "Research on near-threshold energy-efficient processor design based on timing error resilience," Doctor Thesis, Zhejiang University, Hangzhou, China, 2017. (in Chinese)

[20] W. Jin, "Research on ultra-low power pvt tolerant circuits design techniques," Doctor Thesis, Shanghai Jiao Tong University, Shanghai, China, 2017. (in Chinese)

[21] Z. M. Sun, "Near-threshold sram failure estimation and error-tolerant design for convolutional neural networks," Master Thesis, Southeast University, Nanjing, China, 2019. (in Chinese)

[22] Z. G. Wei, "Research on fault tolerance of cache at near-threshold voltage," Master Thesis, Wuhan University of Technology, Wuhan, China, 2018. (in Chinese)

[23] S. Borkar, "Exascale computing - a fact or a fiction?," IEEE International Parallel and Distributed Processing Symposium, Cambridge, USA, pp. 3-3, 2013.

[24] S. Jain, S. Sapatnekar, J. P. Wang, et al., "Computing-in-memory with spintronics," Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, pp. 1640-1645, 2018.

[25] D. Patterson, T. Anderson, N. Cardwell, et al., "Intelligent ram (iram): chips that remember and compute," IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, USA, pp. 224-225, 1997.

[26] Y. Kang, W. Huang, S. M. Yoo, et al., "Flexram: toward an advanced intelligent memory system," IEEE International Conference on Computer Design, Montreal, Canada, pp. 192-201, 1999.

[27] Z. Y. Yu, X. Y. Zeng and S. J. Wei, Microprocessor Design: Architecture, Circuit and Implementation. Science Press, Beijing, China, 2019. (in Chinese)

[28] P. A. Tsai, C. P. Chen and D. Sanchez, "Adaptive scheduling for systems with asymmetric memory hierachies," IEEE/ACM International Symposium on Microarchitecture (MICRO), Fukuoka, Japan, pp. 641-654, 2018.

[29] F. Y. Chen and H. B. Tan, "Research on the architecture of processing in memory towards computation," High Performance Computing Technology, No. 260, pp. 1-9, 2019. (in Chinese)

[30] S. Jeloka, N. B. Akesh, D. Sylvester, et al., "A 28nm configurable memory (tcam/bcam/sram) using push-rule 6t bit cell enabling logic-in-memory," IEEE Journal of Solid-State Circuits, Vol. 51, No. 4, pp. 1009-1021, 2016.

[31] S. Aga, S. Jeloka, A. Subramaniyan, et al., "Compute caches," IEEE International Symposium on High Performance Computer Architecture (HPCA), Austin, USA, pp. 481-492, 2017.

[32] C. Eckert, X. W. Wang, J. C. Wang, et al., "Neural cache: bit-serial in-cache acceleration of deep neural networks," ACM/IEEE International Symposium on Computer Architecture (ISCA), Los Angeles, USA, pp. 383-396, 2018.

[33] M. Kang, S. K. Gonugondla, A. Patil, et al., "A multi-functional in-memory inference processor using a standard 6t sram array," IEEE Journal of Solid-State Circuits, Vol. 53, No. 2, pp. 642-655, 2018.

[34] S. C. Li, D. M. Niu, K. T. Malladi, et al., "Drisa: a dram-based reconfigurable in-situ accelerator," IEEE/ACM International Symposium on Microarchitecture (MICRO), Boston, USA, pp. 303-314, 2017.

[35] Q. Li, J. Zhong and X. Li, "Memory management mechanism for hybrid memory architecture based on new non-volatile memory," Acta Electronica Sinaca, Vol. 47, No. 3, pp. 664-670, 2019.

[36] NVIDIA, (2020). *What is in-network computing?* [Online]. Available: https://zhuanlan.zhihu.com/p/166266347. (in Chinese)

[37] F. Zheng, H. L. Li, H. Lv, et al. "Cooperative computing techniques for a deeply fused and heterogeneous many-core processor architecture," Journal of Computer Science and Technology, Vol. 30, No. 1, pp. 145-162, 2015.

[38] J. Y. Huang, R. R. Puli, P. Majumder, et al., "Active-routing: compute on the way for near-data processing," IEEE Symposium on High-Performance Computer Architecture (HPCA), Washington, DC, USA, pp. 674-686, 2019.

[39] B. Abali, B. Blaner, J. Reilly, et al., "Data compression accelerator on ibm power9 and z15 processors," ACM/IEEE International Symposium on Computer Architecture (ISCA), Valencia, Spain, pp. 1-4, 2020.

[40] E. Choukse, M. B. Sullivan, M. O'Connor, et al., "Buddy compression: enabling larger memory for deep learning and hpc workloads on gpus," ACM/IEEE International Symposium on Computer Architecture (ISCA), Valencia, Spain, pp. 926-939, 2020.

[41] Z. Y. Wang, S. Ma, L. B. Huang, et al., The Principle and Design of Networks on Chip. China Machine Press, Beijing, China, 2016. (in Chinese)

[42] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," Proceedings of the 38th Design Automation Conference (DAC), Las Vegas, USA, pp. 684-689, 2001.

[43] A. Hemani, A. Jantsch, A. Postula, et al., "Network on chip: an architecture for billon transistor era," IEEE NorChip, pp. 1-8, 2000.

[44] L. Benini and G. De Micheli, "Powering networks on chips: energy-efficient and reliable interconnect design for socs," International Symposium on Systems Synthesis (ISSS), Montreal, Canada, pp. 33-38, 2001.

[45] C. C. Feng, "Research on key techniques of bufferless router for network-on-chip", Doctor Thesis, National University of Defense Technology, Changsha, China, 2012. (in Chinese)

[46] S. L. Liu, T. S. Chen, X. X. Feng, et al., "Imr: high-performance low-cost multi-ring nocs", IEEE Transactions on Parallel and Distributed Systems, Vol. 27, No. 6, pp. 1700-1712, 2016.

[47] F. Alazemi, A. Azizimazreach, B. Bose, et al., "Routerless network-on-chip," IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, Austria, pp. 492-503, 2018.

[48] J. W. Wang, "Research of key issues on three dementional network on chip", Doctor Thesis, Nanjing University, Nanjing, China, 2012. (in Chinese)

[49] D. Masoud, E. Masoumeh, L. Pasi, et al., "Cmit-a novel cluster-based topology for 3d stacked architectures," IEEE International Conference on 3D System Integration, Munich, Germany, 2010.

[50] F. Li, C. Nicopoulos, Richardson, et al., "Design and management of 3d chip multiprocessors using network-in-memory," IEEE/ACM International Symposium on Computer Architecture (ISCA), Boston, USA, pp. 130-141, 2006.

[51] D. Park, S. Eachempati, R. Das, et al., "Mira: a multi-layered on-chip interconnect router architecture," IEEE/ACM International Symposium on Computer Architecture (ISCA), Beijing, China, pp. 251-261, 2008.

[52] J. Kim, C. Nicopoulos, D. Park, et al., "A novel dimensionally-decomposed router for on-chip communication in 3d architecture," IEEE/ACM International Symposium on Computer Architecture (ISCA), San Diego, CA, USA, pp. 138-149, 2007.

[53] C. C. Feng, Z. H. Lu, A. Jantsch, et al., "A 1-cycle 1.25 ghz bufferless router for 3d network-on-chip," IEICE Transactions, Vol. 95-D, No. 5, pp. 1519-1522, 2012.

[54] M. Briere, B. Girodias, Y. Bouchebaba, et al., "System level assessment of an optical noc in an mpsoc platform," Design, Automation & Test in Europe Conference & Exhibition (DATE), Nice, France, pp. 1-6, 2007.

[55] C. Batten, "Building manycore processor-to-dram networks using monolithic silicon photonics," IEEE Micro, Vol. 29, No. 4, pp. 8-21, 2009.

[56] A. Shacham, K. Bergman and L. P. Carloni, "On the design of a photonic network-on-chip," International Symposium on Networks-on-Chip (NOCS), Princeton, USA, pp. 53-64, 2007.

[57] K. H. Mo, Y. Y. Ye, X. W. Wu,et al. "A hierarchical hybrid optical-electronic network-on-chip," IEEE Computer Society Annual Symposium on VLSI, Lixouri, Greece, pp. 327-332, 2010.

[58] S. Ma, "Research on the Key Techniques of Routing Algorithm and Flow Control Optimizations for Cache-Coherent Networks-on-Chip", Doctor Thesis, National University of Defense Technology, Changsha, China, 2012. (in Chinese)

[59] L. Q. Cheng, N. Muralimanohar, K. Ramani, et al., "Interconnect-aware coherence protocols for chip multiprocessors," IEEE/ACM International Symposium on Computer Architecture (ISCA), Boston, USA, pp. 339-351, 2006.

[60] N. Eisley, L. Peh and L. Shang, "In-network cache coherence," IEEE/ACM International Symposium on Microarchitecture (MICRO), Orlando, USA, pp. 321-332, 2006.

[61] N. Agarwal, L. Peh and N. K. Jha, "In-network snoop ordering (inso): snoopy coherence on unordered interconnects," IEEE International Symposium on High Performance Computer Architecture (HPCA), Raleigh, USA, pp. 67-78, 2009.

[62] N. Agarwal, L. Peh and N. K. Jha, "In-Network Coherence Filtering: Snoopy coherence without broadcasts," IEEE/ACM International Symposium on Microarchitecture (MICRO), New York, USA, pp. 232-243, 2009.

[63] N. Enright Jerger, "SigNet: Network-on-chip filtering for coarse vector directories," Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, pp. 1378-1383, 2010.

[64] B. K. Daya, C. H. O. Chen, S. Subramanian, et al. "SCORPIO: A 36-Core Research Chip Demonstrating Snoopy Coherence on a Scalable Mesh NoC with In-Network Ordering," International Symposium on Computer Architecture (ISCA), Minneapolis, USA, pp. 25-36, 2014.

[65] Q. Hu, P. Liu, M. C. Huang, et al., "Exploiting transmission lines on heterogeneous networks-on-chip to improve the adaptivity and efficiency of cache coherence", International Symposium on Networks-on-Chip (NOCS), Vancouver, Canada, pp. 1-8, 2015.

[66] J. Hennessy, D. Patterson, "A new golden age for computer architecture: domain-specific hardware/software co-design, enhanced security, open instruction sets, and agile chip development," Turing Lecture in International Symposium on Computer Architecture (ISCA), Los Angeles, 2018.

[67] D. Patterson, "50 years of computer architecture: from then mainframe cpu to the domain-specific tpu and the open risc-v instruction set," IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, USA, pp. 27-31, 2018.

[68] R. I. D. Tullsen, "Heterogeneous computing," IEEE Micro, Vol. 35, No. 4, pp. 4-5, 2015.

[69] N. P. Jouppi, C. Young, N. Patil, et al., "In-datacenter performance analysis of a tensor processing unit," ACM/IEEE International Symposium on Computer Architecture (ISCA), Toronto, Canada, pp. 1-12, 2017.

[70] X. Y. Liang, Ascend AI Processor Architecture and Programming Principles and Applications of CANN. Tsinghua University Press, Beijing, China, 2019. (in Chinese)

[71] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, Sixth Edition. Morgan Kaufmann Publishers, Cambridge, USA, 2019.

[72] O. Temam, "The rebirth of neural networks," ACM/IEEE International Symposium on Computer Architecture (ISCA). Saint-Malo, France, pp. 349-349, 2010.

[73] NVDIA, (2018). *NVIDIA Turing GPU Architecture* [Online]. Available: http://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf.

[74] Y. G. Bao, (2021). *After multi-core, what is the development direction of CPU?* [Online]. Available: https://www.zhihu.com/question/20809971. (in Chinese)