Survey: Scalable Applications and Scalable Systems Postprint

Authors: Sun Ninghui, An Mingyuan, Bao Yungang, Cao Zheng, Chen Fei, Chen Huan, Jiang Xiaoxiao, Li Qiang, Jin Yi, Liu Xingkui, Tan Guangming, Tu Dengbiao, Wang Dawei, Wang Jie, Wang Kai, Wang Wendi, Wang Yang, Xing Jing, Xu Jianwei, Yuan Qingbo, Zhang Wenli, Zou Ming

Date: 2016-11-02T00:00:00+00:00

Abstract

Scalable computer systems have been increasingly widely deployed across various domains. These applications often exhibit scalability requirements, yet the characteristics of such scalable applications vary considerably. Over the past two decades, system platforms for scalable applications have proliferated, each offering distinct advantages. Consequently, evaluating the degree of suitability between a class of applications and a particular system platform has become a critical concern for users. This paper presents a survey and analysis of scalable applications and scalable systems, and proposes several reference factors for assessing the matching degree between applications and system platforms. Additionally, this paper explains and analyzes some hot new terms recently introduced in the industry, comparing their similarities and differences. The objective of this paper is to help readers deeply understand the characteristics of scalable applications and scalable systems, assist users in selecting appropriate platforms to improve application efficiency and resource utilization, and inspire researchers to further explore system platform technologies that address new application requirements.

Full Text

Preamble

Vol. 7 No. 1
Information Technology Letter

Review: Scalable Applications and Scalable Systems

Ninghui Sun, Mingyuan An, Yungang Bao, Zheng Cao, Fei Chen, Huan Chen, Xiaoxiao Jiang, Qiang Li, Yi Jin, Xingkui Liu, Guangming Tan, Dengbiao Tu,

Dawei Wang, Jie Wang, Kai Wang, Wendi Wang, Yang Wang, Jing Xing, Jianwei Xu, Qingbo Yuan, Wenli Zhang, Ming Zou

Abstract

Scalable computer systems have found increasingly widespread applications across various domains, and these applications often exhibit scalable demands, yet the characteristics of these scalable applications differ significantly. Over the past two decades, numerous system platforms for scalable applications have emerged, each with distinct advantages. Evaluating the suitability of a particular application for a specific system platform has become a key concern for users. This paper provides a comprehensive review and analysis of scalable applications and scalable systems, proposing several reference factors for evaluating the matching degree between applications and system platforms. Additionally, it explains and analyzes recently popular new terminology proposed by the industry, comparing their similarities and differences. The purpose of this paper is to help people deeply understand the characteristics of scalable applications and scalable systems, assist users in selecting appropriate platforms to improve application efficiency and resource utilization, and inspire researchers to further explore system platform technologies that adapt to new application requirements.

Keywords: scalable applications, scalable systems, matching, new terminology

1 Introduction

With the development of computer technology, scalable systems have been increasingly applied in scientific research, education, industry, the Internet, and transaction processing, playing a crucial role in improving productivity. As multi-core technology becomes mainstream, scalable applications are gradually becoming the dominant application type. Numerous scalable applications have their own distinct characteristics, yet there is currently no good method for characterizing application features. Scalable computing platforms are also diverse, such as Symmetric Multi-Processing (SMP), Non-uniform Memory Access (NUMA), Massively Parallel Processing (MPP), and clusters. Their technological development paths differ, and no single platform can satisfy all applications. This diversity poses significant challenges for both users and system developers. For users, selecting the most suitable machine among various computing platforms and manufacturers' promotions is quite troublesome. For example, traditional metrics that evaluate computing platform performance using floating-point operation speed are too simplistic, and some benchmarks cannot reflect the true characteristics of applications. Currently, there is a lack of standards for platform selection. System platform developers also struggle to make substantial technological improvements without a deep understanding of application characteristics.

This paper presents a review of scalable applications and systems. First, it uses

three dimensions—application scenario, mathematical model, and resource usage characteristics—to describe and classify applications. Second, it analyzes the features of each typical platform through classification. Through this analysis of application and platform characteristics, the paper also proposes a method for evaluating the matching degree between applications and platforms.

Recently, with the development of the Internet, the industry has proposed several new terms related to scalable applications, such as cloud computing, grid computing, and utility computing. These concepts are both related and distinct, easily causing confusion. This paper will elaborate on these concepts in detail and compare their similarities and differences.

The remainder of this paper is organized as follows: Chapter 2 describes the method for classifying applications; Chapter 3 introduces the classification of platforms; Chapter 4 describes the method for evaluating the matching between applications and platforms; Chapter 5 explains some emerging terminology; Chapter 6 discusses future trends and challenges.

2 Classification of Scalable Applications

2.1 Classification Method

Many methods can be used to classify applications in the computer field. This paper adopts a three-dimensional clustering method. As shown in Figure 1 [Figure 1: see original paper], humans generally follow this process when using computers to solve real-world physical problems: 1) abstract the physical problem and create a mathematical model; 2) design computer-implementable algorithms based on the mathematical model; 3) implement the solution using a computer with a certain architecture; 4) simulate and test the original problem using the implemented system; 5) present the results in some intuitive form; and 6) validate or revise the original mathematical model based on the results. Typically, the mathematical model and algorithms are closely related to the physical problem being solved and are independent of specific computer implementation methods, representing the essence of the problem. However, once the problem is implemented on different computer architectures, it exhibits different resource usage characteristics. Therefore, we separate the algorithm model from its specific implementation and characterize and classify applications through a three-tuple of "application scenario—mathematical model and algorithm—resource usage characteristics." Based on the different domains that applications target, we divide application scenarios into three major categories: scientific and engineering computing applications, Internet applications, and database applications.

2.2 Scientific and Engineering Computing Applications

With the rapid development of computer technology, computation has become one of the three pillars of modern scientific research, alongside theory and experiment. Its application scope covers various fields in modern society, including

industry, agriculture, transportation, healthcare, and education. The core of scientific and engineering computing is to study how to use existing mathematical and logical theories to characterize practical problems and how to effectively solve them using computers. It is a direct product of applied mathematics and computer science development.

2.2.1 Feature Description Method Scientific and engineering computing involves numerous application domains and runs on platforms with huge differences, resulting in significant variations in workload characteristics across different applications on different platforms. Additionally, various optimization strategies adopted to improve computational efficiency, enhance system reliability, and reduce implementation overhead make applications exhibit even more complex runtime differences. Therefore, we divide application characteristics into two categories: (1) static characteristics related to algorithms; and (2) dynamic characteristics related to runtime resource overhead. Algorithmic characteristics directly affect implementation overhead, computational intensity, runtime efficiency, and many other parameters, representing the core features of an application. The algorithm process needs to be mapped to limited computational resource space, generating numerous runtime characteristics. Therefore, we classify scientific and engineering computing applications by algorithmic model, characterize algorithmic features through data representation and load balancing, and characterize resource usage through three metrics: computational intensity, data access patterns, and execution efficiency.

We use the 13 computational models proposed by UC Berkeley as a reference [2], replacing the Combinational Logic problem with Scalar computation, merging Graphical Models and Graph Traversal problems, and adding a separate Partial Differential Equation model, also representing different application computational features with 13 algorithmic models, as shown in Column 1 of Table 2 -1. The benefits of this classification are: 1) it facilitates extracting and analyzing common characteristics among different application workloads, aiding in the development of computational library functions; 2) it benefits communication and discussion among experts from different fields; and 3) since it is not bound to low-level implementation, it greatly improves the design and optimization freedom of computational models.

2.2.2 Algorithmic Model Characteristics In Table 2-1, the metrics directly related to algorithmic models are data representation and load balancing characteristics. If the data representation form of an algorithmic model is regular and the access pattern is predictable, it is classified as Structured; otherwise, it is Unstructured. For example, dense matrix data is typically stored as a linear array, with access patterns following row-major or column-major order. In contrast, sparse matrix data is usually represented in irregular forms such as Compressed Sparse Row (CSR) or Block Compressed Sparse Row (BCSR), with access patterns showing asymmetry due to different positions of non-zero data. For load balancing characteristics, if the parallel implementation of a

computational model requires no or only minimal overhead to achieve balanced computation among nodes, it is marked as "Yes" ; otherwise, it is marked as "No." For instance, the Spectral Computation model' s load balancing among compute nodes can be satisfied by adjusting the number of data points assigned to each node. The Backtrack/Branch-and-Bound model may experience load imbalance issues where some compute nodes meet pruning conditions early and finish computation prematurely while others perform deep searches. It should be noted that marking "No" only indicates a load imbalance at the algorithmic level; through subsequent platform-specific optimizations, practical applications can still achieve load balancing. Table 2-2 presents the analysis results of typical scientific and engineering computing applications from reference [3] using the above algorithmic models.

2.2.3 Resource Usage Characteristics Application resource usage characteristics include four metrics across three aspects: computational intensity, data access characteristics, and efficiency. The first three metrics measure the degree to which algorithm implementation is constrained by resources, while the last metric comprehensively measures resource usage efficiency.

Algorithmic model computational intensity is divided into compute-intensive and non-compute-intensive categories. For example, in Table 2-1, Column 4 marks "Yes" for algorithmic models including Dense Matrix, Partial Differential Equations, and Fast Fourier Transform. The unified characteristic of these models is that their compute-to-memory-access ratio and compute-to-communication ratio are higher than other models marked as "No." If an application contains one or more such compute-intensive algorithmic models, the computing unit is likely to become the bottleneck of the entire application.

Data access characteristics are divided into local and remote categories, as shown in Columns 5 and 6 of Table 2-1. In many current scenarios, the memory wall is the main constraint on application performance. Data access locality, local memory bandwidth, and throughput become major factors affecting data access. The communication intensity and complexity of scalable application algorithmic models also directly affect execution overhead and implementation complexity. For local access characteristics, we describe the spatial characteristic of memory usage using total data volume and characterize temporal characteristics as either bandwidth-limited or latency-limited. For example, in Table 2-1, the Structured Grid algorithmic model involves large amounts of data migrating from one computation point to another between each computation step, requiring substantial data interaction, so its spatial characteristic is "High." However, the migration speed of a single data point does not significantly impact overall runtime speed, so its temporal characteristic is bandwidth-limited. For remote access characteristics, we describe spatial characteristics using whether global data communication is required and characterize temporal characteristics as bandwidth-limited or latency-limited. Using the Structured Grid algorithmic model as an example again, the aforementioned data point migration only oc-

curs between neighboring nodes, so Column 6 of Table 2-1 describes its spatial characteristic as "Neighbor Nodes"and its temporal characteristic as bandwidth-limited. For algorithmic models such as Fast Fourier Transform and MapReduce, due to global communication operations like Broadcast and AllReduce, their remote access spatial characteristic is "Global Nodes."

The overall execution efficiency of an algorithmic model is measured using three levels: "Low," "Medium," and "High." For example, the Dense Matrix algorithmic model has both compute-intensive and load balancing characteristics, corresponding to "High" execution efficiency. Although Spectral Computation also has compute-intensive and load balancing characteristics, it requires global data communication, so its execution efficiency is "Medium."

Table 2-3 presents the implementation strategies of some typical computing platforms in terms of large computation volume, local memory access, interconnection efficiency, and overall efficiency, corresponding to the last four columns of Table 2-1.

The above is only an approximate classification. Actual applications can still vary significantly due to influences from libraries, operating systems, compilers, and other factors. It is important to emphasize that while improving resource-related metrics, we should pay more attention to enhancing resource usage efficiency, which will become a key factor in determining the success of future supercomputer systems [11].

2.3 Internet Applications

The Internet is a global network of computers that communicate with each other using a common language. With the rapid development of the Internet, human society has entered a network information era centered on the Internet, which will play an increasingly important role in human politics, military, culture, and economy.

2.3.1 Classification Method Using the three-tuple method, Internet applications can be divided into two categories based on scenario: Internet-based scientific computing and Internet-oriented information services. Algorithmic models are mainly distinguished by resource distribution and control methods, and the number of online user participants. Resource distribution and control methods include: centralized control server-client mode and distributed peer-to-peer (P2P) mode. The number of online user participants refers to the number of simultaneous online participants in a specific network application. For example, the number of online participants for web search is 1, while for online chat it is 2. Application resource usage characteristics are discussed separately for client-side and server-side.

1. Internet-based Scientific Computing Applications The main working method of Internet-based scientific computing is that the project party di-

vides large computing tasks into small pieces (work units), distributes them to volunteers via the Internet for computation, and volunteers return their computation results to the project party' s server after completion. The characteristics of this application type are: 1) The problem and algorithm themselves have excellent parallelism, and huge problems can be well divided into work units suitable for personal computer scale that can be computed independently; 2) Interaction between participating personal computers during processing is infrequent. Additionally, to adapt to the Internet environment, this application type has added fine-grained checkpointing and experimental result data verification technologies. Fine-grained checkpoints are needed because when personal users need to use their computers, the running scientific computing application should immediately yield CPU resources, and when the personal computer becomes idle, the scientific computing application resumes from some intermediate point, a process that may occur very frequently. Experimental result data verification is needed to prevent online cheating. Internet-based scientific computing applications can be further subdivided based on the physical problems they solve: climate analysis, bioinformatics, physical chemistry, and mathematics. Examples include SETI@home [15] for searching for extraterrestrial life and Folding@home [16] for protein folding computation.

2. Internet-oriented Information Service Applications Table 2-4. Internet-oriented Information Services

Category	Subcategories
Information Portal	Portals, blogs, BBS
Information Retrieval	Web search, map search, image search, video search, digital libraries, file sharing
Network Storage	Network drives, online photo albums, unlimited capacity email
Network Multimedia	Video on demand, audio on demand, online TV, online radio
E-commerce	Online banking, online shopping, stock trading, B2B
Instant Messaging	Email, QQ, MSN, Skype, Fetion

Internet-oriented information services can be summarized and integrated into seven major categories based on the services provided: information portals, information retrieval, network storage, network multimedia, e-commerce, online gaming, and instant messaging. Each major category includes several subcategories, as detailed in Table 2-4.

2.3.3 Algorithm-based Classification Based on the application scenario classification, applications can be more specifically divided into eight quadrant

spaces according to resource distribution and control methods and the number of online user participants, as shown in Figure 2 [Figure 2: see original paper]. When the number of online users is 1, only information services using client/server mode exist, including information portals, information retrieval, e-commerce, and network storage. An important characteristic of this application type is that it does not require high client-side resources and can be implemented using thin clients. When the number of online users is 2, some instant messaging software appears in peer-to-peer mode, such as Skype, while some online games and other instant messaging software appear in client/server mode. When the number of online users ranges from dozens to thousands, information services mainly include client/server mode online games, some network multimedia, and peer-to-peer mode network multimedia applications. Internet-based scientific computing only appears when the number of users reaches thousands or even tens of thousands.

2.3.4 Resource Usage Characteristics Table 2-5 presents the client-side resource usage of Internet-based scientific computing applications, while Table 2-6 shows the server-side resource usage of Internet-oriented information service applications.

2.4 Database Applications

Database is one of the most important and widely used applications in the computer field. Databases are an indispensable component of modern enterprises, and in some industries, databases even occupy a core position, such as banking, aviation, telecommunications, and retail. With a long history and wide usage, database applications are numerous and can be classified according to application scenarios, algorithms, and resource usage characteristics.

2.4.1 Classification of Database Application Scenarios Database applications can be divided into data management, data analysis, and data mining based on scenario. The purpose of data management is to manipulate existing data, with basic operations being insert, query, update, and delete. Online Transaction Processing (OLTP) is the main application form. The purpose of data analysis is to obtain information. As data volume increases, people hope not only to effectively manage data but also to obtain information to support decision-making. Therefore, data warehouses and Online Analytical Processing (OLAP) have been introduced into the database field. Data Cube and Top-K queries are typical OLAP applications. The purpose of data mining is to discover new patterns and knowledge from data. Most data mining work focuses on discovering frequent items, classification, and clustering analysis. Additionally, data streams have attracted attention in recent years. Queries on data streams emphasize timeliness more, and since data streams often involve massive data, applications on data streams focus mainly on online analysis and mining.

2.4.2 Main Algorithms in Database Domain Database applications have a wide variety of algorithms. We can use commonly used ones to analyze application characteristics. Common algorithms in the database domain include: basic relational algebra operations, grouping, sorting, building tree indexes, building histograms, set operations, and graph partitioning (agglomeration). Basic relational algebra operations include selection, projection, and Cartesian product. Grouping and sorting appear everywhere in database applications and often become the foundation for other algorithms, so their performance should be highly valued. Building indexes in databases can often significantly improve query performance. B-tree indexes and R-tree indexes and their variants are the two most widely used tree indexes. Histograms are used not only for data analysis but also for query optimization. Since histograms contain accurate statistical information, they are widely used in stream data processing. Common histograms include equi-depth and equi-width histograms. Set operations and graph partitioning (agglomeration) are also widely used in database applications. Sets and graphs are the most basic and powerful tools for people to understand data.

2.4.3 Resource Usage Characteristics Database applications differ significantly from other applications in resource usage. Since the data volume managed by databases is often much larger than memory, almost all database applications are I/O-intensive rather than compute-intensive. From a resource usage perspective, they can be distinguished by characteristics such as sensitivity to I/O bandwidth, sensitivity to I/O latency, and degree of parallelism. The I/O requirements reveal how an application accesses data. For example, highly concurrent transaction processing, with effective index support, has random I/O access patterns and is sensitive to I/O latency. Periodic sales record analysis, however, often involves sequential data scanning and is therefore sensitive to I/O bandwidth. Generally, database system resources are limited, and the cost of migrating data is substantial. Therefore, the degree of parallelism in database applications can significantly affect performance. To improve parallelism in database applications, effective data organization and partitioning are often required.

3 Classification of Scalable Systems

Computers have been widely used in various fields. Since the first electronic computer was created in 1946, hundreds of different architectures have emerged over decades of development. This section introduces and classifies existing scalable platforms, describing the key technological points in software and hardware for each type of high-performance computer.

3.1 Classification Method

Definitions based on system scale, such as minicomputers, mainframes, and supercomputers, represent the simplest classification of high-performance computers. However, the emergence of a new system is truly driven by applications

and user requirements. Different architecture systems are suitable for one or more applications. General-purpose platforms are designed from the outset to support a wide range of applications by balancing processor, memory, and interconnection designs. Special-purpose platforms are constructed for specific applications, with all aspects of design optimized for one type of application or even one algorithm.

Based on this distinction, this paper classifies existing platforms according to the architecture of general-purpose platforms and the application types of special-purpose platforms, and provides detailed analysis of the technical characteristics of several typical machines.

3.2 General-Purpose Platforms

The evolution of general-purpose platform architectures is the result of continuously adapting to new application requirements. Therefore, general-purpose platforms are classified by architecture and introduced along a time dimension: vector machines, symmetric multi-processing architecture, ccNUMA, massively parallel processing, and clusters. This section describes the history, typical systems, and key technological points of each platform.

3.2.1 Vector Machines A vector processor is one that can process an entire vector with a single instruction. Vector machines represent a milestone in supercomputing. For nearly 20 years from the mid-1970s to the early 1990s, vector machines dominated the supercomputing field. The earliest vector machines were TI' s TI-ASC and Control Data' s STAR-100. These processors had very powerful vector operation capabilities—for example, ASC could perform matrix multiplication with a single simple instruction. However, they only sold a few units and did not achieve commercial success, mainly because their scalar operation performance was poor, inferior to contemporary scalar machines. Moreover, these vector machines only showed advantages when running highly vectorized programs, which greatly limited their applications. The truly successful vector machine was the CRAY-1, introduced in 1976. This machine not only had powerful vector operation capabilities but also outperformed contemporary scalar machines in scalar operations. CRAY-1 opened the golden age of vector machines. Until the early 1990s, vector machines outperformed microprocessors in almost all aspects, including clock frequency, functions per cycle, bandwidth, and I/O, making them the undisputed 霸主 of supercomputing. Several Japanese companies, such as NEC, Fujitsu, and Hitachi, also launched their own vector machines during this period.

Starting in 1992, microprocessors caught up with and surpassed contemporary vector machines in clock frequency. People gradually began using microprocessors to build parallel supercomputers. Moreover, absolute peak performance was no longer the only metric for supercomputers; people began focusing on computing power per dollar. Consequently, the once-dominant vector machines finally gave way to parallel systems composed of microprocessors, such as SMP,

ccNUMA, and clusters. However, vector machines did not completely disappear. In certain specific domains, vector machines still have advantages, such as some scientific computing and image processing applications. In particular, the idea of vector processing has been widely used—most multimedia instructions in microprocessors are used to process vectors (although these instructions are called Single Instruction Multiple Data, SIMD).

NEC has maintained significant investment in vector machine research. Its SX series represents a relatively successful supercomputer based on vector processors. The SX-9, released in October 2007, is the latest member of the SX series and is currently the world' s fastest supercomputer based on vector processors, with a theoretical peak floating-point performance of up to 839 TFlops, primarily targeting scientific computing applications such as weather forecasting, fluid dynamics, and environmental simulation.

The main innovation of the SX-9 system is its newly developed CPU. This CPU contains four independent vector units, each capable of completing 8-way 64-bit floating-point operations per cycle. With a clock frequency of 3.2 GHz, a single CPU' s floating-point performance exceeds 100 GFLOPS. A single SX-9 node consists of 16 CPUs forming an SMP system with computing power reaching 1.6 TFlops. Each CPU has a memory bandwidth of 256 GB/s. A single node supports 1 TB of shared memory space accessed in a Uniform Memory Access (UMA) manner. SX-9 nodes are connected by an ultra-high-speed switch with bidirectional 256 GB/s throughput between nodes. The SX-9 system has good scalability, supporting up to 512 nodes, with peak computing capability reaching 819 TFlops and maximum storage space of 512 TB.

3.2.2 Scalable Symmetric Multi-Processing Architecture After vector machines dominated the high-performance computer market for over a decade, their cost became increasingly high while performance improvements became more difficult. As microprocessor performance continued to improve, Carnegie Mellon University successfully developed a shared-memory multiprocessor system called C.mmp based on the popular DEC PDP-11 minicomputer. This system connected 16 PDP11/40 processors to 16 shared memory modules through a crossbar switch. Subsequently, bus protocols very suitable for SMP architecture emerged, and UC Berkeley extended the bus protocol to propose a solution for the cache coherence problem. Since then, the shared-memory multiprocessor path pioneered by C.mmp has become increasingly popular. Now, this architecture is widely used in the server and desktop workstation markets.

Various server vendors have representative models based on SMP architecture, such as SGI' s Power Challenge XL series parallel computers, Compaq' s Alphaserver 84005/440, and HP' s HP9000/T600. IBM recently launched the Power 595 SMP server supporting up to 64-way processors, using 5.0 GHz POWER6 processor chips with dual memory controllers, utilizing point-to-point interconnect for inter-core communication, supporting up to 4 TB of memory per server and four memory operations per clock cycle, with total memory band-

width exceeding 1.3 TB/s. IBM Power 595 supports multiple operating systems, from high-end UNIX (AIX, IBM i) to Linux (SUSE, Red Hat), can configure up to 254 virtual partitions, and provides active partition migration functionality. It can be applied to enterprise data centers requiring large-scale transaction processing and high Reliability, Availability, and Serviceability (RAS) for critical business database applications.

In SMP systems, system resources are shared by all CPUs in the system, and workloads can be evenly distributed across all available processors. SMP architecture based on point-to-point direct connection technology generally only scales to 4-8 way, such as AMD Opteron. Scalable SMP requires more complex interconnection and switching technologies to connect more CPUs. Since all processors share bus bandwidth, SMP systems have relatively poor scalability, with the number of processors generally fewer than 128. SMP systems can be built into ultra-large-scale computer systems through cluster technology to extend their processing capabilities.

3.2.3 ccNUMA Although SMP architecture simplifies the programming model, as the number of processors increases, the effective bandwidth utilization for memory access becomes increasingly low, and scalability is greatly limited. The emergence of Non-uniform Memory Access (NUMA) architecture partially solved the scalability problem. Since NUMA is a distributed memory structure, to ensure cache coherence among processors, people proposed the ccNUMA architecture with cache coherence protocols. In 1991, Stanford University' s DASH project team completed the world' s first truly operational ccNUMA parallel machine, Dash. The ccNUMA structure did not change the SMP programming model—programmers still organize code as on a regular PC—but system scalability improved significantly. At the same time, this architecture can fully utilize the principle of program execution locality, offering the advantages of simple programming and high execution efficiency. Therefore, since the ccNUMA structure was proposed, it has been a research hotspot, and many server providers have developed and launched ccNUMA servers, with SGI being a faithful developer of ccNUMA servers.

SGI launched the classic ccNUMA system Origin 2000 in 1996. At that time, the 128-processor configuration of Origin 2000 set world records in both SpecCrate_fp95 and SpecCrate_int95 benchmarks. In the November 1997 Top500 list, Origin 2000 series parallel machines occupied 102 positions, demonstrating the brilliant achievements of ccNUMA architecture in supercomputing history. SGI' s latest ccNUMA server is the SGI Altix 4700. The Altix 4700 server has a typical ccNUMA architecture, uses the 4th generation NUMAlink high-performance network, can be configured with up to 512 compute sockets, and can be expanded to a maximum of 128 TB of global memory. Unlike many supercomputers that require specialized operating systems, Altix 4700 can be managed using general-purpose operating systems such as SUSE Linux Enterprise Server. This makes Altix 4700' s design, management, and development

costs more acceptable to users than other supercomputers.

Due to its inability to compete with massively parallel processing in scalability, ccNUMA architecture computers have gradually faded from computing capability rankings as parallel processing scales have grown rapidly. However, ccNUMA architecture' s efficient dedicated network, simple programming environment, and respectable computing capabilities can still meet medium-scale scientific computing requirements.

3.2.4 Massively Parallel Processing Massively parallel processing architecture is the most commonly used architecture for high-performance computers. It has the following characteristics: (1) computers using massively parallel processing architecture contain multiple types of nodes, where a single node may not be a complete computer, and different nodes use operating systems of different scales; (2) they use dedicated interconnection networks, offering the best scalability compared to other architectures.

IBM' s Blue Gene series and Cray' s XT series high-performance computers are typical massively parallel processing systems. Cray' s XT4 high-performance computer uses AMD' s Opteron dual-core processors. The controller connects directly to the processor on one side and builds a 3D torus network with other nodes through a 6-way interface on the other side. The SeaStar2 dedicated interconnection network provides high bandwidth and low latency for communication. The file system uses the parallel object-oriented Lustre. Compute nodes use a lightweight Linux operating system, while service nodes use a complete Linux operating system. System management software provides a unified system image for the upper layer. The Portals communication library supports MPI 2.0 and provides various special operations.

3.2.5 Clusters A cluster system is a collection of multiple interconnected independent computers. These computers can be single-processor or multiprocessor systems (PCs, workstations, SMP, or ccNUMA), with each node having its own memory, I/O devices, and operating system. To users and applications, a cluster is a single system that can provide a low-cost, high-performance environment.

In 1997, a 16-node cluster based on Pentium II processors was launched, costing only $50,000 yet providing one billion floating-point operations per second, while purchasing a parallel machine with equivalent capability at that time would cost ten times as much. UC Berkeley' s NOW system was also an early workstation cluster, consisting of hundreds of Sun Ultra workstations integrated into 19-inch chassis, using various interconnection methods such as Myrinet, ATM, and terminal concentrators. Each node had 512K cache, 128M memory, and two 2.3G hard drives. With cluster development, commercial networks for inter-node communication such as Infiniband and SAN storage networks gradually emerged, making clusters increasingly popular and representing a growing proportion of high-performance computers.

The Ranger system developed by Sun is a typical cluster with a computing peak of 579.4 TFlops, currently ranking fourth in the Top500. Its nodes are SMP blade servers using AMD' s Barcelona CPUs, running the Linux operating system, with nodes interconnected using commercial Infiniband networks forming a fat tree topology, and using the Lustre parallel file system for storage.

3.2.6 Summary of General-Purpose Platforms Table 3 -1. Technical Points of General-Purpose Platforms

Platform Name	Architecture	Processor	System Controller	Operating System	Programming Model
NEC SX-9	Vector	Vector Processor (SIMD)	Ultra-high bandwidth parallel multi-layer crossbar switch	Super-UX, compatible with Unix System V	Message passing + Vector instructions
IBM Power 595	SMP	General-purpose processor	Bus or crossbar switch	High bandwidth, low latency	Single OS, message passing, shared memory
SGI Altix 4700	ccNUMA	General-purpose processor	Distributed shared memory with cache coherence	High-performance dedicated network	Single OS, message passing, shared memory
Cray XT4	MPP	General-purpose processor	Non-shared high-speed cache	Lightweight Linux on compute nodes, full Linux on service nodes	Message passing
Sun Ranger	Cluster	General-purpose processor	Non-shared	Commercial high-performance network, management network, storage network	Mainstream OS, single system image management, parallel file system, checkpointing

3.3 Special-Purpose Platforms

In the history of computer development, many special-purpose systems have emerged, such as Lisp machines and database accelerators. Since the performance improvement speed of special-purpose systems was far lower than Moore' s Law, most systems were short-lived. However, with computer development, power consumption has become an important factor restricting computer scale. In this context, special-purpose platforms that can achieve higher performance per watt have gradually gained attention and development again. To achieve balance in functionality, performance, and power consumption, hybrid computing platforms combining special-purpose and general-purpose components have also increasingly appeared.

Existing special-purpose platforms are often used for molecular dynamics simulation and lattice quantum chromodynamics simulation. The introduction of special-purpose platforms will also focus on these two application types. At the same time, some special components not belonging to specific platforms will also be introduced in this section. Each platform and component will be summarized using the three-tuple method of "special-purpose—targeted application —technical points."

3.3.1 Molecular Dynamics Simulation Molecular dynamics simulation is widely used in life sciences, geophysics, and other fields. The main special-purpose platforms are MDGRAPE-3 and Anton.

1. MDGRAPE-3 MDGRAPE-3, also known as the "Protein Explorer," was developed by Japan' s RIKEN Institute. Its main goal is to provide high-precision visualization for drug development and conduct large-scale simulations of large proteins and chromosome groups. As shown in Figure 3 [Figure 3: see original paper], its structure consists of a host PC cluster and special-purpose engines.

In molecular dynamics simulation, most computation time is spent on non-bonded forces (e.g., Coulomb and van der Waals forces). The special-purpose engine is only responsible for calculating non-bonded forces, while other computations are completed by the host. The system was released in June 2006, consisting of 5,325 MDGRAPE-3 chips, each with speed exceeding 165 Gflops and peak computing capability reaching the petaflop level. The MDGRAPE-3 chip is specially designed for molecular dynamics simulation, with its structure shown in Figure 4 [Figure 4: see original paper]. The chip contains 20 force calculation pipelines. This dedicated pipeline consists of 3 subtractor units, 6 adder units, 8 multiplier units, and 1 function evaluation unit. When calculating Coulomb forces, it can perform approximately 33 equivalent operations per cycle. The 20 pipelines can be virtually used as 40 pipelines. The j-particle storage component saves coordinates of j-particles in 32,768 body models, totaling 6.6 Mbit, eliminating the need for external memory. Additionally, the special-purpose engine uses a scalable 2D hyper-crossbar topology for interconnection.

2. Anton Anton was developed by David E. Shaw Research, with the main goal of simulating millisecond-level life processes for protein prediction and new drug development. The system contains 512 nodes and uses fully custom-designed ASIC chips to implement all computations required for molecular dynamics simulation.

As shown in Figure 5 [Figure 5: see original paper], Anton's High-Throughput Interaction Subsystem (HTIS) implements the most time-consuming non-bonded force calculations, while the Flexible Subsystem implements other molecular dynamics simulation computations. The flexible subsystem embeds general-purpose processors, making the system programmable and able to flexibly implement various algorithms. The entire system uses 3D Torus topology interconnection, routers support multicast, a push-based communication mechanism, and supports compressed communication for sparse data structures, enabling priority transmission for specific data packets. Additionally, its memory system can support accumulation and synchronization operations.

3.3.2 Lattice Quantum Chromodynamics (LQCD) Simulation Through lattice quantum chromodynamics simulation, humans can achieve quantitative understanding and prediction of rich hadron physics phenomena—from nuclear physics phenomena on Earth to matter forms in the early universe, from microscopic material structures to galaxy structures at cosmic scales—and ultimately discover and establish the fundamental laws of nature. The main special-purpose platforms for LQCD simulation are apeNext and QCDOC.

1. apeNext apeNext is a machine released in 2003 as part of the APE project, including 4,096 nodes with a computing peak of 20 TFlops. The APE project started in 1988 with the goal of accelerating LQCD applications, conducted through collaboration between INFN (Italy), DESY (Germany), and the University of Paris-Sud (France). apeNext adopts an architecture similar to MDGRAPE-3, using a general-purpose plus special-purpose mode, with special components accelerating the most time-consuming LQCD operations.

The apeNext ASIC chip is shown in Figure 6 [Figure 6: see original paper]. This chip can implement complex double-precision floating-point operations, with arithmetic units implementing pipelined multiply-add operations and providing 256 128-bit wide registers suitable for floating-point operations. Multiple chips use a dedicated high-performance network interconnection with a 3D Cubic mesh topology, offering good scalability. At the software level, it uses a fully customized operating system and programming environment.

2. QCDOC QCDOC stands for QCD On a Chip, a single-chip QCD processor developed through collaboration among Columbia University, RIKEN (Japan), Brookhaven National Lab (BNL), UKQCD, and IBM. QCDOC was released in 2004 with a peak performance of 10 TFlops. Its structure also adopts a general-purpose plus special-purpose design. As shown in Figure 7 [Figure

7: see original paper], unlike apeNext, it integrates a general-purpose processor core (IBM PowerPC440) for peripheral interface control and component interaction. At the same time, 4 MB of DRAM is integrated on-chip for high-bandwidth memory access. Its custom special-purpose module is called SCU, implementing QCD-specific logic. Multiple QCDOC chips have distributed shared memory and use dedicated high-speed network interconnection. This network supports chained block/strided transmission, multicast, barrier synchronization, and global reduction operations, thus achieving excellent global synchronization operation performance. The network uses a 6D Torus topology, offering good scalability. At the software level, the system uses single-process Unix with a customized programming environment.

3.3.3 Special-Purpose Components In some domains, hybrid platforms combining special-purpose and general-purpose components are increasingly appearing. These platforms are often built by adding accelerator components. Currently, widely used accelerator components include IBM Cell, Nvidia GPU, Clearspeed ASICs, and Field-Programmable Gate Array (FPGA) components.

IBM Cell, Nvidia GPU, and Clearspeed ASICs can be considered as one category. They can target multiple application domains but are most suitable for floating-point-intensive applications. For example, Clearspeed's latest CSX700 ASIC can provide 96 GFlops of double-precision computing capability. IBM' s Roadrunner uses IBM Cell as an accelerator component to accelerate streaming computing applications.

FPGA components have also begun to be applied in high-performance computing, with the typical representative being the Cray XT5h system. FPGA components can fully utilize hardware parallel features to flexibly implement some applications and improve their performance, especially suitable for programs with regular execution processes.

3.3.4 Summary of Special-Purpose Platforms Table 3-2. Technical Points of Special-Purpose Platforms

Platform Name	Target Application	Architecture	Special Design	Memory	Interconnection
MDGRAPE-3	Molecular Dynamics	General-purpose processor + Special-purpose processor	Physical force calculation pipeline	On-chip SRAM	2D hyper-crossbar topology

Platform Name	Target Application	Architecture	Special Design	Memory	Interconnection
Anton	Molecular Dynamics	Fully custom ASIC	Hardware logic implementation of molecular dynamics algorithms	Small cache	3D Torus topology, push-based communication, priority transmission for special packets
apeNext	LQCD	General-purpose + Special-purpose	Complex double-precision operations, pipelined multiply-add	Multi-wide register file	3D cubic mesh topology
QCDOC	LQCD	General-purpose core + LQCD-specific logic	High-performance floating-point operations	On-chip embedded memory	6D Torus topology, multicast & barrier, global reduction

4 Evaluating the Matching Degree Between Scalable Applications and Systems

The previous sections described scalable applications and platforms. For a given application, finding the most suitable platform is quite complex. Application requirements come from three aspects: programmers, administrators, and the application itself. It is difficult for the platform itself to consider every aspect thoroughly, requiring certain trade-offs. This chapter proposes a method for evaluating whether an application matches a platform. First, application requirements are refined to list the basic characteristics of platforms. Then, the impact of each platform characteristic on application requirements is presented. By considering these impacts and combining them with user requirement priorities, we can determine the required platform characteristics to meet user needs, and then examine which platforms satisfy these characteristics to select a matching platform. The development of high-performance computer architecture is a process of continuously adapting to application requirements. This section uses the above evaluation method to analyze the reasons for the development of

mainstream high-performance computer architectures, identifies the most suitable applications for clusters and massively parallel processing through analysis of the past 7 TOP500 lists, and provides examples of matching processes for four different types of applications.

4.1 Application Requirements and Platform Characteristics

4.1.1 Application Requirements From the perspective of a single application, platform requirements have three sources: programmers, administrators, and the application itself. Each requirement affects the final platform selection. The requirement list is shown in Table 4 -1.

Table 4-1. Application Requirements

Source	Requirements
Programmer	Easy to program, easy to port, easy to debug
Administrator	Scalability, ease of use, ease of management, availability
Application	Machine price, maintenance cost, supporting equipment
Performance	Response time, throughput, efficiency

4.1.2 Platform Characteristics A complete platform consists of hardware and software components. Hardware includes architecture-related parts and component characteristics, while software includes management software, operating systems, and development environments. Platform characteristics are shown in Table 4-2.

Table 4-2. Platform Characteristics

Category	Characteristics
Architecture	Processors, memory, I/O, interconnection
Components	Special-purpose, general-purpose
Management	Job/system management, single image
OS	Fault tolerance, single image
Development Environment	Programming models, debugging

4.1.3 Impact of Platform on Application Each specific platform characteristic affects certain application requirements. As shown in Figure 8 [Figure 8: see original paper], architecture characteristics affect TCO, SUMA, and system saturation performance requirements. Special-purpose or general-purpose components affect TCO on one hand and indirectly affect system saturation performance on the other. Management software and operating systems only affect SUMA, while development environment characteristics affect application requirements for programming portability.

4.2 Development of High-Performance Computer Architecture

The development of high-performance computer architecture is primarily driven by applications. Old architectures gradually disappear because they are unsuitable for application requirements, while new architectures that better adapt to applications become mainstream. Once new application requirements emerge, originally mainstream architectures give way. The relationship between architecture development and application requirements is shown in Figure 9 [Figure 9: see original paper].

Vector machines emerged in the early 1960s and were very suitable for compute-data-intensive applications like Computational Fluid Dynamics (CFD) [18]. However, they were gradually replaced by SMP/NUMA architectures due to their weakness in TCO. SMP/NUMA architectures share address space but have poor scalability, so they were replaced by massively parallel processing. Massively parallel processing has excellent performance and scalability, but because it uses special-purpose components (networks, operating systems) leading to increased costs, clusters emerged. Clusters use entirely commercial components, have low manufacturing and maintenance costs, low entry barriers, and have replaced massively parallel processing to become mainstream. However, facing various challenges in high-performance computing, clusters are gradually becoming inadequate and must confront three major challenges: First, the three-low challenge, namely low power consumption, low footprint, and low price (we call this LPC—Low Price/power/proportion Computer); Second, the Very Large Scale Parallelism (VLSP) challenge, which brings difficulties in system reliability, interconnection network scalability, parallel algorithm design, parallel programming models and languages due to ultra-large-scale parallelism; Third, the Productivity challenge, which requires improving the output capability of supercomputer centers as public infrastructure, including key factors that constrain actual application efficiency such as communication and synchronization performance, I/O performance in multi-user environments, requirements for fault isolation and resource virtualization, management technologies to improve system utilization and reduce total cost of ownership, and simplifying user application development and debugging. Therefore, we recently proposed a new HPP architecture [19] to address these deficiencies in clusters.

Table 4-3 mainly compares the characteristics of four types of high-performance computer architectures from three aspects: high-performance computing, utility computing, and energy consumption. High-performance computing includes processing granularity, communication performance, I/O, reliability, manageability, scalability, and programmability. Utility computing includes commercialization degree, virtualization granularity, resource sharing, and isolation. Power consumption is listed as an independent item.

Table 4-3. Architecture Characteristic Comparison

Architecture	High-Performance Computing	Utility Computing	Power Consumption
Vector Machine	Fine granularity, high communication performance	Low commercialization	High
SMP/ccNUMA	Medium granularity, shared memory	Medium commercialization	Medium
Massively Parallel Processing	Coarse granularity, high scalability	Low commercialization	Medium
HPP	Message passing, ultra-large scale	High commercialization, fine virtualization	Low

4.3 Statistical Analysis of Application-Platform Matching

This section uses statistics from the past 7 TOP500 lists (3,500 machine instances) to determine the most suitable applications for massively parallel processing and clusters. The rationale is: if many machines of a certain architecture are used for a specific application, we can conclude that this application and architecture are suitable; if there are few or no such applications, it can be used as evidence of mismatch between application and architecture.

As shown in Figure 10 [Figure 10: see original paper], since 2005, machines in clusters used for semiconductor, geophysics, and finance applications have ranked at the top, especially financial applications showing a continuous growth trend, with more than 70 machines used for such applications each year in recent years.

Figure 11 [Figure 11: see original paper] shows the two most used applications on massively parallel processing high-performance computers: weather and climate forecasting, and research. In recent years, more than 20 massively parallel processing high-performance computers have been used for research.

Figures 10 and 11 show the most installed applications in clusters and massively parallel processing architectures, respectively. But what about the usage of the most used applications in clusters on massively parallel processing, and vice versa? If in both cases the other side' s usage is minimal, it can be concluded that the application is more suitable for clusters or massively parallel processing. Table 4-4 shows the total installed numbers of the five most widely used applications across 7 TOP500 lists. It can be seen that applications suitable for clusters are rarely used on massively parallel processing machines, while applications suitable for massively parallel processing are not used at all on clusters. We can infer that semiconductor, geophysics, and finance are most suitable for clusters, while weather and climate research and research applications are most suitable for massively parallel processing.

Table 4-4. Comparison of Cluster and Massively Parallel Processing Applications

Application	Cluster	Massively Parallel Processing
Semiconductor	High usage	Low usage
Finance	High usage	Very low usage
Geophysics	High usage	Low usage
Weather & Climate	Very low usage	High usage
Research	Very low usage	High usage

4.4 Other Examples

This section illustrates the application-platform matching process with examples. Table 4-5 shows application requirements for platforms across four different application types: finance, weather and climate forecasting, molecular dynamics, and quantum simulation. Finance mainly uses the Monte Carlo method; weather and climate forecasting mainly uses the finite element method; molecular dynamics includes multiple algorithms, with the spatial domain algorithm presented here; quantum simulation mainly uses the Fast Fourier Transform algorithm. These methods have different platform requirements.

Table 4-5. Application Requirements for Platforms

Application	Algorithm	Communication Pattern	Parallelism	Programming Model
Finance (Monte Carlo)	Large collective communication	Medium	Very large	OpenMP
Weather & Climate (Finite Element)	Large point-to-point	Few	Very large	MPI
Molecular Dynamics (Spatial)	Large collective communication	Few	Very large	OpenMP
Quantum Simulation (FFT)	Large collective communication	Few	Very large	MPI

Finance application users emphasize low cost, making clusters the obvious best choice when other conditions are equivalent. Weather and climate forecasting

and molecular dynamics applications are more suitable for massively parallel processing due to their high communication performance requirements. Quantum simulation applications are special—they use the Fast Fourier Transform algorithm [23], where communication time is comparable to computation time, resulting in very low parallel efficiency. For example, the efficiency of FFTW' s high-dimensional parallel algorithm is generally only 20-30%. None of the current mainstream architectures can adapt well, leading to the emergence of high-performance computers that can efficiently support the Fast Fourier Transform algorithm, such as the Earth Simulator and Blue Gene/L. On the Earth Simulator with 5,104 processors, the measured Fast Fourier Transform performance is 35.61 TFlops, equivalent to 87.2% of its theoretical peak of 40.96 TFlops. Blue Gene/L' s three-dimensional Fast Fourier Transform communications-intensive kernel is customized based on the torus network structure, and a 128×128×128 MPI FFT can scale to 8,192 processors.

5 New Terminology Related to Scalable Systems

Currently, the industry has proposed several new concepts related to scalable systems that are easily confused, such as cloud computing, Data Intensive Scalable Computing (DISC), grid computing, and utility computing. This section explains these concepts and compares the differences among these computing modes.

5.1 Computing Modes

The initial computing mode for computers was mainframe/terminal mode. With the popularization of networks, many new computing modes emerged. The mainstream new computing modes mainly include three types: client/server mode, master/slave mode, and peer-to-peer mode.

Client/server represents a service relationship between two computer programs, where one program as a client sends service requests to another program, which as a server processes the requests after receiving them. In network environments, the client/server mode provides a convenient interaction method for cross-node distributed operations. The client/server mode is very common in computer applications and has become the mainstream design method in network computing. Today, most commercial programs are written in client/server mode, including TCP/IP applications on the Internet. The browser/server mode used in Internet applications fixes the client program as a unified Web browser.

Master/slave is also an interaction mode in network computing, where one device or process (the "master") controls one or more devices or processes (the "slaves"). Once the master/slave relationship is established, the control direction remains from master to slave.

Peer-to-peer mode means that each party (node, peer) in the computation has the same capability and can simultaneously have both client and server functions. Each party provides part of the global system resources. There is no

central coordinator or central database, and no unit can have a complete system perspective. Nodes obtain needed information through interaction.

5.2 Cloud Computing

Cloud computing is a new computing mode. Google calls large-scale computer clusters on the Internet "clouds." Google and IBM launched a "cloud computing" cooperation project in October 2007 to improve the data processing capability of the "cloud." The driving force behind this new computing mode is that enterprises or individuals will no longer need to install large amounts of application software on their computers but will access large-scale, on-demand customized services through Web browsers, called "cloud services." This will fundamentally change the user experience. In cloud computing mode, user-required applications do not run on personal computers, mobile phones, or other terminal devices but run on large-scale server clusters on the Internet. User-processed data is not stored locally but saved in data centers on the Internet. Enterprises providing cloud computing services are responsible for managing and maintaining the normal operation of these data centers, ensuring sufficient computing power and storage space for users. Users only need to access these services anytime, anywhere, using any Internet-connected terminal device. In the Internet-centered era, computing and data are moving toward centralization, existing in different data centers— "clouds" —but there will still be much computing running on terminals, which Microsoft calls Cloud-Client, indicating that clients should also share tasks suitable for terminal devices.

5.2.1 Characteristics of Cloud Computing Dr. Kai-Fu Lee, Google' s Vice President, believes cloud computing has four important characteristics: (1) massive data storage in the cloud; (2) countless software and services placed in the cloud; (3) all built on various standards and protocols; (4) accessible through various devices.

For ordinary users, cloud computing brings a better user experience. First, cloud computing provides the most reliable and secure data storage center, eliminating worries about data loss and virus invasion. Second, cloud computing has the lowest requirements for user-side devices and is most convenient to use. Additionally, cloud computing can easily achieve data and application sharing among different devices. Cloud computing provides almost unlimited possibilities for using networks, almost unlimited space for storing and managing data, and almost unlimited computing power for completing various applications.

5.2.2 Business Model Compared with the software industry in the PC era, the business model in the "cloud" era is completely different. Internet-based software has no version concept. For example, Google' s Gmail and Gdocs adopt the "forever beta" software development model. This model stays close to users, does not produce learning curves that jump with product lifecycles, can achieve countless online updates and upgrades daily, and mainly achieves profit through

advertising. Traditional software industries represented by Microsoft treat software as products rather than services, with Windows operating systems, Office, and other software fixedly installed on clients, using a paid business model to achieve profit.

5.2.3 Cloud Computing Cases The greatest value of cloud computing lies in allowing enterprises to focus on their own business and technological innovation while outsourcing required infrastructure resources (such as servers, storage, networks, etc.) to computing clouds, using and paying for them on demand, thereby reducing the total cost of ownership for enterprise applications. For example, Amazon provides network application infrastructure services Elastic Compute Cloud (EC2) and Simple Storage Service (S3). The EC2 computing cloud service uses Redhat and Xen-based solutions for virtual machine management. Each EC2 computing unit is a virtual server. The most basic level computing unit capability is equivalent to a 32-bit machine with a 1.0-1.2 GHz Intel Xeon processor, 1.7 GB memory, and 160 GB storage, costing $0.1 per hour. Similarly, storing 1 GB of data on S3 costs only $0.16 per month. Users can dynamically use any number of computing units and storage according to application needs, paying for actual usage without worrying about hardware resource maintenance and management costs. According to Amazon' s Q4 2007 financial report, Amazon' s EC2 and S3 services are very popular. The bandwidth used by these services in Q4 2007 even exceeded the total bandwidth used by Amazon' s global website in the same period. This means that Internet startups and other companies used more of Amazon's network computing infrastructure than Amazon itself. Currently, 330,000 global developers are engaged in network application innovation on EC2 and S3.

The New York Times provides a free, fully searchable archive service, opening all archives from 1851 to 1922, including 15 million articles. The New York Times' senior software architect Derek Gottfrid outsourced this task to Amazon, using EC2 and S3 to generate 11 million PDF documents from its archive library. A new application software called TimesMachine appeared on the New York Times website.

5.2.4 Cloud Computing Platforms Cloud computing' s Internet services are essentially data-processing-centered services. The success of Internet services largely depends on the scale and quality of provided data, the timeliness of data processing, and the proportion of effective data. The successful experience of Internet companies such as Google, Amazon, eBay, and Salesforce shows that creating new Internet economic models and advancing traditional economies requires revolutionary infrastructure platforms and technologies. Such platforms should be able to support network services for tens of millions or even hundreds of millions of users, efficiently, reliably, and cost-effectively process massive data, and lower the threshold for building new network services and applications. Taking Google as an example, Google' s own infrastructure platform remains a commercial secret, reportedly consisting of about 450,000 servers

distributed worldwide. Google connects these servers into a global computing platform through customized Linux and scheduling software called Global Work Queue, achieving efficient ultra-large-scale data processing at relatively low cost. Its indexed hundreds of billions of web pages have become a source for testing various innovative ideas (e.g., Google Labs). The distributed processing technologies used by Google, such as Google File System [25], MapReduce [26], and BigTable [27], provide strong support for meeting its continuously growing business needs.

5.3 Data Intensive Scalable Computing (DISC)

5.3.1 Background of Data Intensive Scalable Computing The development of science and technology has brought us into the digital age, whose notable characteristic is massive data, referred to as "data flood" in [28]. Massive data is generated from two aspects:

1. **Increasingly rich data sources:** Currently, science, humanities, business, entertainment, and medical activities all need to generate, store, and process large amounts of data. For example, in the scientific domain, the SDSS system attempts to provide optical images for more than 1/4 of space, including three-dimensional graphics of galaxies and stars. This system can generate 200 GB of data daily. In the commercial domain, Wal-Mart needs to record 200 million sold items daily. On the other hand, with Internet popularization, more download services are provided on networks, such as online videos and music, and storing and processing these resources requires massive storage space.

2. **Cheap storage:** Technological development has made storage systems increasingly larger, with storage media evolving from tape to disk to new technologies like optical storage. Currently, a single DVD can store about 10 GB of data. Meanwhile, the price per unit storage has become increasingly lower. According to March 2008 data, the price of server-grade 500 GB hard drives had dropped to 699 yuan. The folk saying that storage price has become "cabbage price" reflects this cheap storage.

These two factors have led to massive data generation. However, the most critical problem caused by massive data is data transmission. A notable characteristic of current massive data operations is that a single interactive operation needs to traverse massive data (TB level), while existing data transmission methods, whether through networks or direct reading from hard disks or optical discs, are inadequate compared to massive data storage. For example, for 1 TB of data, reading from a hard disk takes 2-4 hours, and transmission via gigabit Ethernet also takes about 2 hours—unacceptable for interactive operations.

To solve this problem, people naturally thought of the "parallel" approach, using large numbers of disks and nodes to process this data simultaneously, so that each node needs to process a relatively small amount of data, and the narrow channels between disks and processors can meet requirements. This has given

rise to data intensive scalable computing.

Data intensive scalable computing systems have four major characteristics: (1) massive data, where each operation needs to access large amounts of data to complete, but computing power requirements are not high; (2) problem-oriented programming models, i.e., high-level programming loosely coupled with underlying systems; (3) interactive access, especially query-oriented; (4) high fault tolerance requirements, needing high-availability mechanism support.

From the above description, it is clear that data intensive scalable computing systems face two major challenges: how to effectively distribute and store data so that data operations are completed within nodes as much as possible, thereby avoiding or minimizing data movement; and how to efficiently manage these parallel nodes.

5.3.2 Challenges Faced by Data Intensive Scalable Computing Systems Data intensive scalable computing systems consist of hundreds or thousands of nodes, posing enormous challenges in design, management, programming, and usage. This section analyzes in detail the problems that need to be solved to address these challenges.

1. Design Considerations

- **How to design a balanced system:** For a given data intensive scalable computing application, the system's requirements for storage, interconnection, and computing capabilities are relatively fixed. Under this premise, how to design a balanced system that fully utilizes resources in all aspects is a consideration.
- **How to reduce energy consumption:** As system scales expand, the energy consumption problem of large-scale parallel systems becomes increasingly apparent. The latest Top500 has begun to pay attention to energy consumption, adding Mflop/s/Watt metrics to evaluation standards [17]. Google has established a dedicated team to study this problem, and Microsoft and Yahoo have also begun to address it.
- **How to meet hardware requirements:** Whether to choose cheap PCs or high-performance components. A Google study shows that current commercial high-performance CPUs are not fully utilized in Google clusters. If functions are trimmed, it would bring price and power consumption improvements. On the other hand, although Google' s search engine and MapReduce applications demonstrate the feasibility of using cheap PCs to implement a data intensive scalable computing system, is this applicable to other applications?
- **How to measure cost-effectiveness:** Faced with different data intensive scalable computing systems and different applications, can a cost-effectiveness measurement standard be given to quickly provide cost-effectiveness parameters for an application on a system? Such a standard would be very useful for both system designers and users.

- **How to support heterogeneous components:** Traditional supercomputers are oriented toward scientific computing, which is tightly coupled and synchronization-intensive, so systems generally use homogeneous nodes. However, data intensive scalable computing systems are different and can consider using heterogeneous components to build heterogeneous nodes, providing convenience for future system upgrades and expansion.

2. System Management and Runtime Support

- **Large-scale system management:** Data intensive scalable computing systems often consist of thousands of nodes, making efficient management of such massive resources very important.
- **High fault tolerance requirements:** In large-scale systems, component failures are very common phenomena. Therefore, erroneous or temporarily unavailable data needs to be shielded at the system level. How to design a highly reliable system based on component failure and recovery times is a consideration.
- **Resource sharing and isolation:** Data stored in data intensive scalable computing systems is often used by multiple users, requiring both shared access to common data and protection of private data.
- **High concurrency data consistency support:** In data intensive scalable computing systems, the same data may be operated on by multiple applications simultaneously, exhibiting high concurrency characteristics. Since data often has multiple replicas, maintaining data consistency among all replicas in the system is necessary.
- **Self-optimization capability:** Traditional high-performance computers are generally oriented toward specific scientific computing. To fully utilize system potential, applications need to understand system hardware characteristics in detail and make corresponding optimizations, often requiring specialized knowledge. Data intensive scalable computing systems differ in hoping that ordinary users can easily develop applications, so system optimization should rely more on compilers or system software.

3. Programming Models

- **Problem-oriented high-level programming models:** Traditional parallel programming models in high-performance computing such as MPI and pthread require understanding of underlying configurations during programming. Data intensive scalable computing systems hope to adopt general, problem-oriented, high-level abstract programming models that allow users to focus only on the application itself without needing to understand underlying configuration information. The mapping between applications and data intensive scalable computing systems should be completed by compilers or system software.
- **Asynchronous coarse-grained parallelism:** Data intensive scalable computing systems require parallel application components to execute asynchronously, meaning there are no strict synchronization requirements between parallel components in terms of start time and duration, and the

success of one parallel component' s execution has no strict relationship with others.

- **Fault tolerance requirements:** In addition to the aforementioned system-level fault tolerance support, the programming model should also provide fault tolerance support.

4. Usage Patterns

- **Interactive applications:** Taking Google' s search engine as a typical example, data intensive scalable computing applications involve frequent user interactions, with each user operation needing to access massive data. Therefore, the coexistence of interactive applications and massive data operations is a challenge faced by data intensive scalable computing systems.

5.3.3 Typical Data Intensive Scalable Computing Systems 1. Google File System (GFS) [30]

- **Massive data support:** Large file chunks and streamlined metadata information. Google File System divides data into 64 MB chunks, with each chunk' s metadata information being less than 64 bytes. In this case, for 100 TB of data, metadata information is less than 100 MB and can be operated in memory.
- **High fault tolerance support:** Main technical measures include: (1) Data replication. Data storage nodes define the number of data replicas based on different security requirements. Metadata storage nodes also have corresponding replicas. (2) System operation logs. Since all operations must go through the metadata server, all operations are recorded on the metadata server side. These logs can be replayed during system failure to restore the system to the state at the time of failure. (3) Data checksums. To ensure the integrity of individual data blocks, each block k carries checksum information.
- **High concurrency and consistency support:** Main technologies include: (1) File-based read/write locks. This allows multiple users to simultaneously add files to the same directory during write operations. (2) Using separated control flow and data flow to ensure data consistency, where control flow is determined by a unique data server, and all data flows use the order specified by this control flow to update data.

2. Google Programming Model MapReduce [31]

Google' s MapReduce programming model originates from functional languages. Google has developed many libraries using this model, promoting it to more and more applications. Its main functions include: (1) Automatically distributing tasks for parallel processing; (2) Tolerating faults; (3) Executing I/O scheduling; (4) Monitoring task status.

Its main features are: (1) Not modifying input data status, always generating new data; (2) Data flow is implicit in program design; (3) Operation order is

irrelevant.

Its main operation flow is shown in Figure 12 [Figure 12: see original paper].

5.3.4 Application Classification Broadly speaking, data intensive scalable computing is applicable to all data-intensive service applications, such as Amazon's pay-per-use cloud computing service system EC2, pay-per-storage-capacity S3 network storage system, and data-intensive computing applications such as natural language processing, gene function prediction, earthquake prediction, astronomical exploration, animation production, and brain behavior analysis. The former has been introduced extensively. Below we mainly introduce data-intensive computing applications (computational category) of data intensive scalable computing.

Common applications in natural language processing include machine translation of languages, whose core is establishing statistical language models and translation algorithms. Language models need to continuously obtain the latest documents from the network and recalculate them, requiring significant storage and computing power. For example, in the 2005 NIST machine translation competition, the Google research group that won first place used 1,000 processors for simultaneous translation.

Gene function prediction mainly compares gene sequences of different organs of the same species or different species to discover information hidden in DNA. As new gene sequences are continuously discovered, the space required to store gene information is also increasing. According to data released by the National Center for Biotechnology Information (NCBI) in August 2006, the world' s gene bank already had 65 billion nucleotide records at that time, and this record was growing at a rate of doubling every 10 months.

Astronomical scientists often use astronomical telescopes to obtain large amounts of image information and then analyze and mine these images to discover new astronomical phenomena. Currently, the Sloan telescope located in New Mexico, USA, can capture 200 GB of astronomical image data daily, and the latest released image set has reached 10 TB.

Animation production requires first creating geometric information of targets, then making targets move through motion capture and force field simulation methods based on models, and finally rendering target systems by adding virtual lighting. The entire process involves processing massive data, and the granularity of data processing directly affects animation production quality. Often, a few seconds of footage requires several hours of processing.

5.3.5 Summary Traditional numerical computing application workload characteristics are: compute-intensive, requiring high CPU floating-point processing capability; frequent data exchange between different CPUs requiring high-bandwidth, low-latency communication networks; requiring large parallel I/O

bandwidth; high hardware reliability requirements needing system-level checkpointing and hardware-level fault tolerance; many concurrent applications requiring complex job management. In contrast, data intensive computing application workload characteristics are completely different: not requiring high CPU floating-point processing capability but high fixed-point processing capability; small global communication volume, large communication granularity, low latency sensitivity; large original data volume requiring large concurrent I/O bandwidth, good data locality during processing; low hardware reliability requirements allowing application-level fault tolerance; few concurrent applications requiring only simple job management. Therefore, scalable computer systems targeting data intensive computing applications, characterized by massive data storage and operation, have significant differences from traditional high-performance computers in system design and programming models, bringing challenges to system design, runtime environment, programming models, and application development.

Google has built a data intensive scalable computing system based on cheap PCs and successfully supported Internet data-intensive applications through GFS file system, MapReduce programming model, and BigTable database technology. However, data intensive scalable computing systems are not just this one application model of Google. Whether all applications can be solved using similar methods is currently the biggest problem for data intensive scalable computing systems. For example, can all data intensive scalable computing applications be implemented using programming models with MapReduce characteristics? Can all data intensive scalable computing applications adopt asynchronous coarse-grained parallelism? Solving these problems requires in-depth understanding of applications and algorithms to form specific complete system solutions for this type of data intensive scalable computing applications.

5.4 Grid Computing

Grid computing was first proposed in the early 1990s by Ian Foster and Carl Kesselmans in their book "The Grid: Blueprint for a new computing infrastructure." The name "grid computing" comes from the power grid, with the idea of making access to various computing resources as convenient as using electricity. This concept emerged with its era background. On one hand, high-performance computing and large-scale commercial applications have higher requirements for computer resources, needing to integrate computing resources from various autonomous institutions to achieve large-scale resource sharing and collaborative work to solve larger-scale complex problems. On the other hand, the development of computer hardware, software, and the Internet has provided necessary conditions for resource sharing and collaborative computing across wide area networks.

5.4.1 Concept The grid integrates and shares geographically distributed computing and storage resources to form a complete collaborative computing envi-

ronment that solves large-scale scientific and commercial problems. Through open standards and protocols, the grid integrates resources from multiple autonomous domains to form a virtual single-image computing environment, allowing users to conveniently use various resources in the virtual computing environment without caring about the specific physical locations of these resources.

It is widely believed that the grid is the third wave of the Internet after the Internet and the Web. On the initial Internet, resource sharing was mainly achieved through telnet remote login to computer nodes, implementing computer-to-computer interconnection. After Web applications were invented, network resource sharing became page-to-page interconnection. In the grid era, the Internet will achieve comprehensive interconnection of all resources.

5.4.2 Implementation Since a grid system needs to span multiple autonomous domains, an open standard is needed to interconnect various autonomous and possibly heterogeneous systems. The Open Grid Services Architecture (OGSA) is a service-oriented grid architecture standard developed by the Global Grid Forum (GGF).

Following the OGSA standard, the international organization Globus Alliance developed the Globus Toolkit for building grid systems. Globus studies key grid computing technologies such as resource management, security, information services, and data management, providing basic mechanisms and interfaces. Currently, most grid projects adopt protocols and services based on the Globus Toolkit [32]. IBM' s IBM Grid Toolbox, SUN' s Sun Grid Engine, and the Institute of Computing Technology' s Vega Grid also provide solutions for building grids and grid applications.

5.4.3 Applications According to Foster and the Globus project team, grid application domains currently mainly include four categories: distributed supercomputing, distributed instrumentation systems, data-intensive computing, and tele-immersion [33].

1. **Distributed Supercomputing:** Connects physically distributed supercomputers through high-speed networks, using middleware to integrate resources from various autonomous systems to form a virtual computing environment with powerful computing capability to solve large-scale scientific problems. Applications such as military simulation and black hole simulation belong to this category.
2. **Distributed Instrumentation System:** Uses grids to manage valuable instrument systems distributed in various locations, providing remote access to equipment, improving instrument utilization, and facilitating user usage. Applications include remote medical care and astronomical telescope sharing.
3. **Data-Intensive Computing:** Processes large-scale data. Data grids focus more on data storage, transmission, and processing. Fields such as

biology, medicine, and Earth observation all generate massive data, and computing in these fields belongs to data-intensive computing.

4. **Tele-immersion:** Provides a virtual reality environment through grids to reproduce history, reflect reality, or visualize various types of computational results. Users can roam freely in the same virtual space through networks, communicate with each other, and interact with the virtual environment to change it. Applications include virtual history museums, collaborative learning environments, and collaborative data visualization analysis.

5.4.4 Relationship Between Grid and Cloud Computing Cloud computing is the development of distributed computing, parallel computing, and grid computing, or the commercial implementation of these computer science concepts [34]. As shown in Figure 13 [Figure 13: see original paper], the grid provides infrastructure for cloud computing implementation in resource management, security mechanisms, and other aspects. In fact, any cloud is supported by grid computing technology [35].

5.5 Utility Computing

To use or provide a computing service, computing resources that support the service must be obtained. Under existing computing resource usage models, this usually means purchasing and owning a certain amount of computing resources. In addition, deploying, managing, and maintaining these computing resources require varying degrees of expertise, which are not concerns for computer users but bring higher total cost of ownership. The concept of utility computing is a commercial model proposed by the information technology industry to solve these problems. Utility computing is also often called on-demand computing. In the utility computing model, computing resources (such as computing and storage resources) are encapsulated, similar to public utilities like electricity, water, and natural gas, and provided as a metered service. Obtaining computing services no longer requires owning computing resources but renting or purchasing resources as needed. Rental costs are measured based on resource usage, similar to public services people are accustomed to using.

5.5.1 Concept Utility computing is a resource sharing and on-demand service application model and the development direction for future data centers. Many current technologies are oriented toward this goal. For example, Web Service is an application-layer on-demand service technology in the Internet application domain, grid computing is an application-layer on-demand service technology in the high-performance computing domain, virtual machine is a computer system resource sharing technology, and partition is a large server resource sharing technology. Currently, there is still a lack of effective support for utility computing at the computer architecture level [40].

Utility computing is not a new concept. In 1961, Turing Award winner John

McCarthy publicly proposed at MIT' s centennial celebration when looking forward to the future development prospects of time-sharing technology: "If the computer I envision becomes the mainstream form of future computers, then perhaps one day computing will become a public facility like the telephone system··· computing utility will become the foundation of a new and important industry" [14]. IBM and other mainframe manufacturers dominated this business model for the following twenty years, providing computing and storage resources to banks and large organizations through data centers worldwide, which also became the early form of utility computing development. In the mid-to-late 1970s, the development of microprocessor technology and the emergence of minicomputers allowed people to obtain computer hardware at much lower prices than before. Computer users tended to purchase their own computing centers instead of buying computing resources from large manufacturers. It was not until the late 1990s that the concept of utility computing regained attention and new development. IBM' s Global Services department' s new definition of utility computing is: "Utility computing is using infrastructure, applications, and business processes on demand through the Internet in a secure, shared, scalable, and standards-based computing environment. Users can access information technology resources as easily as they currently use electricity and water and pay according to usage" [37].

5.5.2 Key Technologies Utility computing provides services by encapsulating computing resources (computing and storage). Users do not need to own resources but rent/purchase them as needed. To implement utility computing, there are three main key technologies:

1. **Resource metering:** The system should be able to reasonably measure resource usage for commercial charging.
2. **Operation and data security guarantees in shared environments:** Supporting shared environment operation is a prerequisite for utility computing to become popular, and data security guarantees are key to its use in production systems.
3. **Resource management:** How to dynamically supply resources for multiple applications. This technology includes three aspects: resources allocated to users need to adapt to changes in the number and types of applications; the system needs to adjust resource allocation to meet users' changing needs during operation; and the system' s resource supply must meet Service Level Agreements (SLA).

5.5.3 Applications Some information technology vendors have begun providing utility computing-like services. In spring 2006, Amazon released Amazon EC2 (Elastic Compute Cloud) [16]. Amazon EC2 uses Xen virtualization technology. Users can select and rent any number of server instances from four different capability levels as needed. Users have a high degree of control over rented server instances and can perform operations such as creation, startup, and shutdown. Sun provides Sun Grid Compute Utility [17] service through

www.network.com, mainly for compute-intensive applications, charging $1 per hour.

5.5.4 Relationship with Virtualization Utility computing practice usually requires support from virtualization technology. Through virtualization technology, storage or computing resources of multiple back-end computers can be aggregated into much more shared physical resources than a single computer. For example, VMware proposed Virtual Infrastructure (see Figure 14 [Figure 14: see original paper]). A Virtual Infrastructure represents all resources in the entire IT environment, aggregating all x86 computers and attached network and storage resources into a unified resource pool, which can greatly improve resource utilization.

5.6 Chapter Summary

The four computing modes introduced in this chapter can be divided into two categories in nature. Data intensive scalable computing can be considered as one category independently, emphasizing large-scale data computing, mainly how to use large-scale computing equipment to process data in parallel. The other three computing modes can be considered as another category, with the commonality of emphasizing resource integration and sharing but different focuses. Utility computing emphasizes resource integration efficiency, focusing more on the technical level; grid computing emphasizes building frameworks, mainly focusing on structural implementation; cloud computing emphasizes providing easy-to-use computing services to users through the Internet, more of a commercial concept.

The proposal of these three concepts aligns with the needs of their times. When utility computing was proposed, resources were scarce, requiring improved resource utilization, so the concept focused on resource usage efficiency. When grid computing was proposed, computing resources were no longer scarce, but software structures that could unite resources were lacking, so the concept focused on building structures. When cloud computing was proposed, there were abundant computing resources and functional software middleware, so the concept focused on popularizing shared computing and software services.

With increasing data processing demands, updated network infrastructure, and popularized computer applications, the future trend is that more and more computing resources will form one or more unified logical wholes through networks to provide various software services to the outside world. Meanwhile, client hardware and software platforms will be simplified, and mobile services will become increasingly perfect. Future computer system structure research should focus on server-side technology, integrating and utilizing loosely coupled structural resources, mainly including resource management, parallel processing, reliability, and other aspects.

6 Future Challenges

This chapter presents some predictions about future development trends of scalable applications and systems and corresponding challenges. Starting from three levels—application, system, and matching—we fully explore unique development trends and potential impacts at each level while clearly revealing their mutual constraints and promotions.

6.1 Application-Level Challenges

1. The emergence of new parallel programming languages cannot "unify the world" in the near future. How to define programming models in ultra-large-scale parallel environments and design corresponding programming languages is a very challenging problem. There have been no fewer than a thousand parallel programming languages, and current MPI, OpenMP, UPC, and Co-array Fortran are far from ideal parallel programming languages. Whether Cray' s Chapel, IBM' s X10, and Sun' s Fortress can rise abruptly is still a question. Some scholars predict that research on parallel programming languages will need at least 20 to 30 years to mature.

2. Large-scale high-precision physical simulation will become the mainstream application on supercomputer platforms with petaflop-level computing power. Scaling existing applications to petaflop-level platforms and developing more petaflop-level applications is a challenging problem. In recent years, there have been some research works in this area. For example, in [3], researchers predicted fields involved in applications suitable for petaflop-level platforms: nuclear fusion simulation, fluid dynamics, astrophysics, high-energy physics, materials science, and gas dynamics. The High Productivity Computer Systems (HPCS) program hosted by DARPA faces application scenarios including interactive weather and ocean prediction, biotechnology, large-scale engineering design simulation, and new weapon design simulation [13].

6.2 System-Level Challenges

1. The combination of special-purpose components and general-purpose platforms will become the mainstream structure of future high-performance computers. Many early computers were special-purpose machines. Only in the 1990s, with semiconductor technology following Moore' s Law, did general-purpose platforms' performance improvement speed exceed that of special-purpose platforms, making them market-dominant. The Top500 ranking based on the Linpack benchmark also illustrates the decline of special-purpose platforms from one side. Today, power consumption has become the most important dimension constraining system performance. How to obtain higher performance per watt is a primary issue that system designers must consider. In this environment, the development of special-purpose platforms has begun to turn around. Compared with general-purpose systems, special-purpose components can accelerate by hundreds or even thousands of

times, breaking free from power consumption constraints with extremely high acceleration ratios. Researchers at Lawrence Berkeley National Laboratory (LBNL) are designing special-purpose platforms for weather simulation [14]. Through modeling and analysis of applications, they conclude that to meet the computational, storage, and I/O capability requirements of next-generation weather simulation, considering power consumption and hardware scale constraints, building corresponding general-purpose platforms is extremely difficult or even impossible. Therefore, if their analysis is correct, special-purpose platforms are the inevitable choice. At the same time, designers are also finding a balance between general-purpose and special-purpose, with hybrid systems continuously emerging, such as IBM Roadrunner. Correspondingly, special-purpose components will play significant roles in future systems. Of course, special-purpose platforms will not replace general-purpose platforms, but all-general-purpose platforms will lose dominance. The integration of the two types of platforms will be the development trend.

2. The cluster technology route lacking innovation and based on commercial components will exit the stage of high-end systems. Accelerating industrialization while ensuring technological leadership is contradictory. Solving the gap between existing advanced technology and actual market demand is a challenging problem. As early as the 1990s, leaders in the U.S. government and industry had already noticed the prospects of clusters. There has always been a contradiction on the high-performance computer development path: on one hand, cluster technology is relatively mature and can meet the vast majority of current commercial computing needs, and general-purpose processors following Moore' s Law still have great performance improvement space; on the other hand, from a long-term perspective, the inherent defects of cluster technology make it unable to meet high-end computing needs. The U.S. High Productivity Computer Systems program has also begun to pay attention to the industrialization transformation of new technologies, requiring companies like IBM and Cray with strong R&D capabilities and substantial government research funding to apply new technologies developed for high-end systems to mainstream market products.

3. Low-power, high-productivity systems will become the mainstream of future high-end high-performance computing. Various low-power design technologies still face many challenges and need to overcome contradictions between low-power and high-productivity goals. Through statistical analysis of Top500 data over the past 15 years, researchers in [11] predict that supercomputers represented by IBM' s Blue Gene series, which improve peak computing speed while reducing system power consumption, are receiving increasing attention and may open a new trend in low-power, high-performance system design and development. As shown in Figure 15 [Figure 15: see original paper], low-power, high-productivity systems (represented by BG/L and BG/P) may replace current clusters to become the mainstream of TOP500 before 2020.

4. Recent developments in general-purpose processor architecture

may diverge. Heterogeneous systems are the future development trend. How to allocate computing tasks to the most suitable computing units will directly affect the entire system' s execution efficiency, while improving processor programming efficiency is also challenging. Cray' s Cascade system tells us that there is no single processor suitable for all applications, which also illustrates the necessity of researching special-purpose processors from one side. However, developing special-purpose computing components for each application is very time-consuming and inefficient. Therefore, extracting common computing features to implement general-purpose processors within a certain range is the most effective solution so far. How to extract common computing features and what extraction methods to use? According to an article in Linux Magazine [12], general-purpose processors can be divided into two categories: computationally predictable, such as GPUs and CELL, which can only efficiently handle certain specific computing tasks with high programming overhead but high execution efficiency; and computationally unpredictable (i.e., traditional CPUs), such as Xeon, Opteron, and Power7, which can complete many computing tasks but are much less efficient than the former category for some computing tasks. The divergent development of these two types of general-purpose processors will become an inevitable trend for some time. We look forward to their mutual promotion and integration, leading to new branches that better combine the advantages of both.

6.3 Challenges for Matching

Productivity will become a new evaluation standard for future systems. Extracting productivity parameters from the combination of numerous applications and different platforms will be a challenge. Due to natural differences, different user tasks, application scenarios, end-user requirements, and workflows will all place different demands on productivity. How to establish a set of measurement metrics is a key factor determining the accuracy of extracted parameters. Generally, high-productivity systems need to meet some prerequisites [13]: load balancing, high reliability, measurable performance, scalable systems, and efficient and easy-to-use software systems.

Furthermore, greater challenges come from the demands of scalable applications like Google and Amazon on scalable platforms. This will stimulate new original innovations in system technology from architecture and CPU technology to operating systems and management technology. We believe that in the near future, scalable applications and scalable systems will achieve considerable development.

References

[1] Michael T. Heath, "Scientific Computing: An Introductory Survey, 2nd Edition" , McGraw-Hill Companies, Inc., July 2002
[2] Krste Asanovic, Ras Bodik, et al., "The Parallel Computing Laboratory at U.C. Berkeley: A Research Agenda Based on the Berkeley View" , Technical

Report No. UCB/EECS-2008-23, March 21, 2008
[3] Leonid Oliker, Andrew Canning, et al., "Scientific Application Performance on Candidate PetaScale Platforms" , IPDPS 2007, Page(s):1-12, March 2007
[4] Steven J. Ludtke, Philip R. Baldwin, Wah Chiu, "EMAN: Semi-automated Software for High-Resolution Single-Particle Reconstructions", Journal of Structural Biology 128, 82-97 (1999)
[5] http://dft.sandia.gov/Socorro/mainpage.html
[6] Taiji, M. Narumi, T. Ohno, Y. Futatsugi, N. Suenaga, A., et al., "Protein Explorer: A Petaflops Special-Purpose Computer System for Molecular Dynamics Simulations" , Supercomputing, Page(s):15-21, Nov. 2003
[7] http://en.wikipedia.org/wiki/IBM_Roadrunner
[8] George Almasi, Calin Cascaval, Jose G. Castanos, et al., "Dissecting Cyclops: A Detailed Analysis of a Multithreaded Architecture", SIGARCH Computer Architecture News, 2003
[9] Adiga, N.R., Almasi, G., et al., "An Overview of the BlueGene/L Supercomputer" , Supercomputing, Page(s):60-68, Nov. 2002
[10] Oliker, L., Carter, J., Wehner, M., Canning, A., Ethier, S., et al., "Leading Computational Methods on Scalar and Vector HEC Platforms" , Supercomputing, Page(s):62-68, Nov. 2005
[11] Hans W. Meuer, "The TOP500 Project: Looking Back over 15 Years of Supercomputing Experience" , http://www.top500.org, January 20, 2008
[12] http://www.linux-mag.com/id/6552
[13] Jack Dongarra, Robert Graybill, William Harrod, Robert Lucas, et al., "DARPA' s HPCS Program: History, Models, Tools, Languages" , ADVANCES IN COMPUTERS 72, ACADEMIC PRESS, JUN-2008
[14] Michael Wehner, Leonid Oliker, John Shalf, "Towards Ultra-High Resolution Models of Climate and Weather" , International Journal of High Performance Computing Applications, Vol. 22, No. 2, 149-165 (2008)
[15] http://setiathome.ssl.berkeley.edu/
[16] http://folding.stanford.edu/English/Main
[17] http://www.top500.org
[18] Kenichi Miura, "CFD Applications and Performance on Fujitsu Vector-Parallel Processing System: VPP300/VPP700" , Parallel CFD in Taiwan, 1998
[19] Ninghui Sun, Kai Li, Mingyu Chen, "HPP: An Architecture Supporting High Performance and Utility Computing" , Chinese Journal of Computers, Vol. 31, No. 9, September 2008
[20] Karsten Decker, "The Monte Carlo Method in Science and Engineering: Theory and Application" , Technical Reports, 1990
[21] Zienkiewicz, O. C., "The Finite Element Method" , McGraw-Hill, 1977
[22] Melvin L. Prueitt, "Computer Simulation of Molecular Dynamics" , University of New Mexico, 1971
[23] R. W. Ramirez, "The FFT: Fundamentals and Concepts" , Prentice-Hall, 1985
[24] http://en.wikipedia.org/wiki/Cloud_computing
[25] Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung, "The Google File System" , Proceedings of the 19th ACM Symposium on Operating Systems

Principles, 2003
[26] Dean, Jeffrey and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters" , Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI 2004)
[27] Chang, Fay, Jeffrey Dean, Sanjay Ghemawat, et al., "Bigtable: A Distributed Storage System for Structured Data" , Proceedings of the 7th Symposium on Operating System Design and Implementation (OSDI 2006)
[28] Tony Hey and Anne Trefethen, "The Data Deluge: An e-Science Perspective", Tech report, Department of Electronics and Computer Science, University of Southampton, 2003
[29] http://www.sdss.org/
[30] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, "Google File System" , SOSP 2003
[31] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters" , OSDI
[32] Globus Toolkit, http://baike.baidu.com/view/1170090.htm
[33] "Current Status of Grid Application Research", http://www.gridhome.com
[34] www.cloudcomputing-china.cn
[35] http://en.wikipedia.org/wiki/Cloud_Computing
[36] McCarthy, J., MIT Centennial Speech of 1961 cited in "Architects of the Information Society: Thirty-five Years of the Laboratory for Computer Science at MIT" , S.L. Garfinkel Ed., MIT Press, Cambridge MA, 1999
[37] M. A. Rappa, IBM Systems Journal, Volume 43, Issue 1, Pages: 32-42, 2004
[38] http://www.amazon.com
[39] http://www.sun.com/service/sungrid/index.jsp
[40] Jeanne W. Ross and George Westerman, "Preparing for utility computing: The role of IT architecture and relationship management" , IBM Systems Journal, Volume 43, Number 1, 2004, pp. 5-19

Author Biographies

Ninghui Sun: Researcher at the Institute of Computing Technology, Chinese Academy of Sciences; Director of the Key Laboratory of Computer System and Architecture, Chinese Academy of Sciences
Mingyuan An: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Yungang Bao: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Zheng Cao: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Fei Chen: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Huan Chen: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Xiaoxiao Jiang: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences

Yi Jin: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Qiang Li: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Xingkui Liu: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Guangming Tan: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Dengbiao Tu: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Jie Wang: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Kai Wang: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Yang Wang: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Dawei Wang: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Wendi Wang: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Jing Xing: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Jianwei Xu: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Qingbo Yuan: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Wenli Zhang: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences
Ming Zou: Graduate student, Institute of Computing Technology, Chinese Academy of Sciences

Note: Figure translations are in progress. See original paper for figures.