International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Analysis of Complex Project through Parallel Computing

Sidhi Raina

[1]*Student, All India Shri Shivaji Memorial Society's Institute of Information Technology, Pune, Maharashtra, India*

ABSTRACT

The purpose of this analysis is to identify the need to speed up the simulation of the complex project. Using the developed parallel algorithm can be said to be efficient in both projects There are low-risk and high-risk, varying number of project variants, but with the greatest impact for complex projects with many different variants. The higher the proportion of parallel the computing part can use more processors more efficiently. High parallelism the calculation includes the following steps: Decompose subtasks, analyse dependencies and Communication between subtasks. To reach your goal, you must first develop parallel lines Analysis of complex project algorithms and speedup results at different initials condition. Parallel development algorithms are suitable for both small number of projects Optionally for complex projects with thousands of variants.

Keywords: Parallel, project, complex, high-risk, low-risk

1. INTRODUCTION

Recent advances in cloud computing research have achieved remarkable results commercial interest in using cloud infrastructure to support commercial use applications and services. However, there are important developments in the area of risk and risk wide commercial acceptance requires reliability reality. In particular, risk management mechanisms should be integrated into the cloud. Infrastructure beyond a best-effort approach to service delivery what current cloud infrastructure follows. Importance of risk management in the cloud computing arose out of the need to support the various stakeholders involved in manufacturing informed decisions about contractual agreements lack of sufficient trust given the uncertainties that may be associated with quality levels, cloud services. Prevent users of cloud services from using cloud technologies. Although the provision of a zero-risk service is not practical, if not impossible, an effective and efficient risk assessment of service provision and consumption, together with the corresponding mitigation mechanisms, may at least provide a technological insurance that will lead to high confidence of cloud service consumers on one side and a cost-effective and reliable productivity of cloud service providers resources on the other side. Risk assessment has been introduced into utility computing such as Grids and clouds either as a general methodology or focusing on a specific type of risk, such as security and SLA fulfilment. However, the aim of this paper is to propose a risk assessment framework for cloud service provision, in terms of assessing and improving the reliability and productivity of fulfilling an SLA in a cloud environment. Based on this framework, a software tool is designed and implemented as a risk assessment related module, which can be integrated into other high level cloud management and control software systems for both end users and Ips.

2. RELATED WORK

2.1 Parallel computing concept in computer system

All semiconductor market domains are converging to concurrent platforms. This trend has certainly led real challenge to develop applications software that effectively uses these concurrent processors to achieve efficiency and performance goals.

2.2 Parallel Computing Framework

In recent years, the field of big data processing has increasingly demanded computer performance for data processing. In order to enable users to use the big data processing platform conveniently and quickly, and put more attention on the writing of parallel computing algorithms, this paper focuses on cloud services, parallel computing, Iron Python and virtualization as the main research content, and uses Python for the client, Service node and computing node module design, using the reflexive and self-observing capabilities of Python, calling library functions to achieve functions that cannot be achieved in other languages. The parallel computing framework based on Python designed in this paper can be easily implemented on the virtual machine of the cloud computing platform, and can efficiently and intelligently distribute computing requests.

3. PROPOSED SYSTEM

The main memory of a parallel computer is either shared memory (shared among all processors) elements of a single address space) or distributed memory (where each processing element is stored) has its own local address space) [9]. Distributed memory means that the memory Logically distributed, but often also physically distributed. Dispersion Shared memory and memory virtualization combine two approaches that involve processing. Element has its own local memory and accesses memory on non-local processors to access. Accessing local storage is generally faster than accessing non-local storage. Parallel computers can be broadly categorized by the state of hardware Supports concurrency. This classification is divided into Basic and compute node. They are not mutually exclusive. For example, a symmetric cluster Multiprocessors are relatively common.

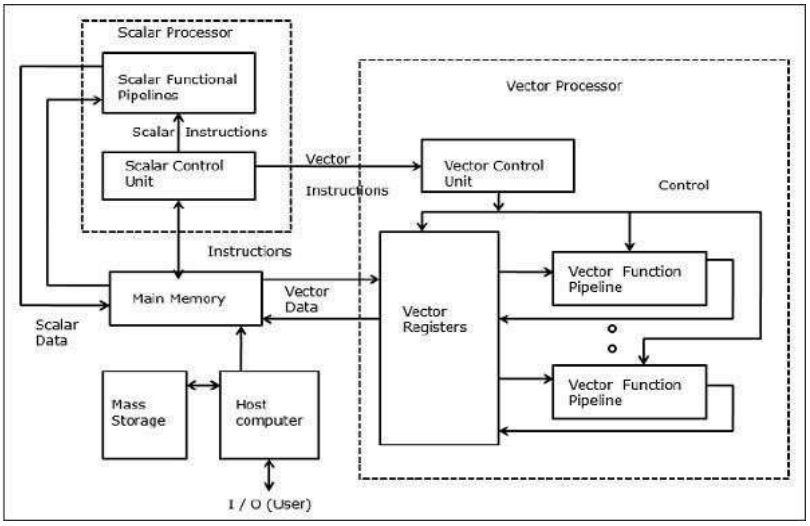

Fig 1 Diagrammatical Design

4. METHDOLOGY

4.1 Amdahl's law and Gustafson's law

Optimally, the acceleration from parallelization is the processing element should halve the runtime and double the second. Halve the runtime. However, few parallel algorithms achieve optimal acceleration. Many they exhibit near-linear acceleration for a small number of processing elements and become flat. Converts to constant values for a number of processing elements. possible acceleration algorithms on parallel computing platforms are originally given by Amdahl's law. Formulated by Gene Amdahl in the 1960s. That's because a small part of the program is it cannot be parallelized, which limits the overall speedup available from parallelization. A programs that solve big math or engineering problems usually consist of: Multiple parallelizable parts and multiple non-parallelizable (sequential) parts. fraction of execution time spent on parts of the program that cannot be parallelized. Am Dahl's law and Gustafson's law assumes that execution time is the sequential part of the program regardless of the number of processors. Amdahl's Law assumes that the problem is solved The size, and thus the total amount of work done in parallel, also does not depend on the number Gustafson's Law states that the total amount of processors is parallel scales linearly with the number of processors.

Fig 2:Graphical Repesentation

4.2 Dependencies

Understanding data dependencies is fundamental in implementing parallel algorithms. No program can run more quickly than the longest chain of dependent calculations (known as the critical path), since calculations that depend upon prior calculations in the chain must be executed in order. However, most algorithms do not consist of just a long chain of dependent calculations; there are usually opportunities to execute independent calculations in parallel. Let Pi and Pj be two program segments. Bernstein's conditions describe when the two are independent and can be executed in parallel. For Pi, let be all of the input variables and Oi the output variables, and likewise for Pj. P i and Pj are independent if they satisfy.

4.3 Memory and communication

The main memory of a parallel computer is either shared memory (shared by all processing elements within a single address space) or distributed memory (each processing element has its own local address space). Distributed memory refers to memory that is logically distributed, but often also means that it is physically distributed. Distributed shared memory and memory virtualization combine two approaches in which processing elements have their own local memory and access memory on non-local processors. Accessing local storage is generally faster than accessing non-local storage. A computer architecture that allows access to all elements of main memory with equal latency and bandwidth is known as a Unified Memory Access (UMA) system. Typically, this can only be achieved through shared memory systems where the memory is not physically distributed. Systems without this property are called Non-Uniform Memory Access (NUMA) architectures. In distributed storage systems, storage access is inconsistent. Computer systems use caches, small fast memories close to the processor that store temporary copies of memory values (nearby in the physical and logical sense). Parallel computer systems have problems with caching where the same value can be stored in multiple locations, which can lead to program execution errors. These computers need a cache coherence system that keeps track of cached values. Remove them strategically to ensure proper program execution. Bus snooping is one of the most common ways to track which values are being accessed (and should therefore be removed). Designing large-scale, high-performance cache coherency systems is a very difficult problem in computer architecture. The result is a shared memory computer architecture.It cannot scale like a distributed storage system. Communication between processors and processor memory is hardwired in a variety of ways, such as shared memory (multiported or multiplexed), crossbar switches, shared buses, or interconnection networks with myriad topologies such as stars, rings, and trees. can be implemented in hardware. , hypercube, fat hypercube (a hypercube with multiple processors in its nodes) or an n-dimensional mesh. Parallel computers based on interconnected networks require some form of routing so that messages can be passed between nodes that are not directly connected. The medium used for interprocessor communication can be hierarchical in large multiprocessor machines.

4.3.1. Classes of parallel computers

Parallel computers can be roughly classified according to the level at which the hardware supports parallelism. This classification is broadly analogous to the distance between basic computing nodes. These are not mutually exclusive; for example, clusters of symmetric multiprocessors are relatively common.

4.3.2 Multicore computing

Main products: Multicore (computing) A multicore processor is a processor that contains multiple execution units (cores) on the same chip. These processors differ from superscalar processors, which can issue multiple instructions per cycle from the instruction stream (thread). In contrast, multicore processors can issue multiple instructions per cycle from multiple instruction streams. IBM's Cell microprocessor, developed for use in the Sony PlayStation 3, is another excellent multi-core processor of his. Each core in a multicore processor can be superscalar. H. Each cycle, each core can issue multiple instructions from the instruction stream. Simultaneous multithreading (best known for Intel's Hyper-Threading) was an early form of pseudo-multicores. A processor capable of simultaneous multithreading has only one execution unit (core), but when that execution unit is idle (such as on a cache miss), it uses that execution unit to process a second thread .

4.3.3 Symmetric multiprocessing

Main products: Symmetric Multiprocessing A symmetric multiprocessor (SMP) is a computer system with multiple identical processors that share memory and are connected via a bus. [10] Bus contention prevents scaling of bus architectures. Therefore, SMPs typically contain up to 32 processors enough Memory bandwidth is available. "

4.3.4 Distributed computing

Distributed Computing Main articles: Distributed Computing A distributed computer (also called a distributed memory multiprocessor) is a distributed memory computing system in which the processing elements are connected by a network. Distributed computing is highly scalable.

4.3.5 Cluster computing

A cluster is a group of loosely coupled computers that work so closely together that in some respects they can be thought of as a single computer. [12] A cluster consists of multiple standalone machines connected via a network. The machines in the cluster don't have to be symmetrical, but load balancing is more difficult if they aren't. The most common type of cluster is the Beowulf cluster. This is a cluster implemented on multiple identical off-the-shelf

computers connected to his local TCP/IP Ethernet network. This technique was originally developed by Thomas Sterling and Donald Becker. Most of the TOP500 supercomputers are clusters.

Fig 3: A Beowulf cluster Main article: Computer cluster A Beowulf cluster

4.3.6 Grid computing

Main products: Distributed Computing Distributed computing is the most common form of parallel computing. It uses computers that communicate over the Internet to tackle specific problems. Due to the low bandwidth and high latency of the Internet, distributed computing is usually only able to deal with massively parallel problems. Many distributed computing applications have been developed, SETI Home and Folding Home being the best known examples. Most grid computing applications use middleware, software that sits between the operating system and applications to manage network resources and standardize software interfaces. The most widely used middleware for distributed computing is Berkeley Open Infrastructure for Network Computing (BOINC). Distributed computer software often uses spare cycles. ”

4.3.7 Massive parallel processing

Main products: Massively Parallel (Computing) A Massively Parallel Processor (MPP) is a single processor.

A computer with many network processors. MPPs share many of the same characteristics as clusters, but MPPs have a dedicated interconnect network (clusters use commodity hardware for networking). MPPs also tend to be larger than clusters, typically with well over 100 processors [15]. With MPP, each CPU has its own memory and a copy of the operating system and applications. Each subsystem communicates with other subsystems over high-speed links. [16] Case by Blue Gene/L, classified as his 4th fastest supercomputer in the world according to the November 2008 TOP500 ranking. Blue Gene/L is a massively parallel processor. Blue Gene/L, his fifth fastest supercomputer in the world according to the June 2009 TOP500 ranking, is an MPP.

4.3.8 General-purpose computing on graphics processing units (GPGPU)

Main products: GPGPU General-purpose computing on Nvidia's Tesla GPGPU Card Graphics Processing Unit (GPGPU) is a very recent trend in computer engineering research. A GPU is a co-processor heavily optimized for computer graphics processing. Computer Graphics processing is dominated by parallel data manipulation, especially linear algebra matrix manipulation. Early GPGPU programs used regular graphics APIs to run programs. However, several new programming languages and platforms are being developed to perform general-purpose computations on GPUs, and both Nvidia and AMD have released programming environments using CUDA and Stream SDK respectively. Other GPU programming languages include Brook GPU, Peak Stream and Rapid Mind. Nvidia too Released specific products for computing in the Tesla series. The Khronos Group, a technology consortium, has published the OpenCL specification, a framework for writing programs that run cross-platform and consist of CPUs and GPUs. AMD, Apple, Intel, Nvidia, etc. support OpenCL.

Fig 4: Nvidia’s Tesla GPGPU card

5. ANALYTICAL/ EXPERIMENTAL WORK EXPLAINATION

Parallel computing refers to the process of dividing a large problem into smaller, independent, often similar pieces that can be executed simultaneously by multiple processors communicating through shared memory. The results are combined upon completion as part of the overall algorithm.

5.1 Comparison

Parallel Computing Distributed computing takes place on a single computer. Multiple computers are involved. Multiple processors execute multiple tasks simultaneously. Multiple computers perform tasks simultaneously. Computers can have shared memory or distributed memory. Every computer has its own memory. Processors communicate with each other through a bus Computers communicate with each other through a network Improves system performance Allows scalability, resource sharing, and helps perform computational tasks efficiently.

5.2 Advantages and Disadvantages

5.2.1 Advantages

- Parallel computing models the real world. The world around us is not serial.
- Save time. Serial computing forces fast processors to work inefficiently.
- Save money. By saving time, parallel computing reduces costs.
- Solve more complex or larger problems.
- Take advantage of remote resources

5.2.2 Disadvantages

- Programming for parallel architectures is a bit more difficult, but with the right understanding and practice you can get started.
- Parallel computing can be used to solve computationally and data intensive problems on multicore processors, but it can affect some of the control algorithms and give poor results. This can also affect system convergence due to possible parallelism.
- Additional costs (that is, increased execution time) are incurred by data transfer.
- Synchronization, communication, thread creation/destruction, etc. These costs can be very high in some cases and even exceed the benefits of parallelization.
- To improve performance, different code adjustments need to be made for different target architectures. • Clusters require better cooling technology.
- High power consumption due to multi-core architecture

5.3 Challenges

Type of parallel computer.

- Problem size and algorithmic complexity.
- Parallel programming languages.
- Processor core type – homogeneous or heterogeneous.
- Number of processor cores

6.APPLICATION

- Numerical Weather Forecasting (NWP) uses mathematical models of the atmosphere and ocean. Obtain current weather observations and process this data with computer models to predict future weather conditions. Use data assimilation to generate results.
- AI is machine or software intelligence. AI systems require massive amounts of parallel computing in which they are used. Four types 1. Image processing 2. Expert systems 3. Natural language processing (NLP) 4. Pattern recognition. • Finite element analysis (FEA) is a numerical technique commonly used for Multiphysics problems. Used to design megastructures such as ships, dams and supersonic jets
- Parallel processing elements are used in FEA because a large number of partial differential equations must be solved simultaneously.

- Weapon Research and Defense Computer clusters are tools used in national defense, used in simulations that show the performance of nuclear weapons at the precise molecular level. Parallel computing is needed to more efficiently authenticate nuclear weapons and accurately represent molecular-scale reactions that occur in milliseconds or thousands of seconds. It is also used in plutonium research to study its behavior under high pressure, and its alloys are used in the production of explosives.

7.CONCLUSION

Parallel computing helps to save the time of computation and solve larger problem over that provided by a single computer. Performance of parallel computing is measured with speed up and efficiency. Based on the results of the research, we conclude that it is advisable to build up the cores on one computing node or the number of computational nodes working on solving the problem simultaneously. At the same time, the number of threads should tend to the number of variants of the project.

8.REFERNCES

[1] Han Wan, Xiaopeng Gao, Xiang Long, Bo Jiang: “Introducing parallel computing concepts in computer system related courses”, IEEE Frontiers in Education Conference (FIE), December – 20

[2] Aidong Fang, Lin Cui, Zhiwei Zhang, Congjiang Chen, Zhuang Sheng, “A Parallel Computing Framework for Cloud Services IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), August -2020

[3]. "Official Neurocluster Brain Model site". Retrieved July 22, 2017.

[4]. Kaku, Michio (2014). The Future of the Mind.

[5]. Gupta, Ankit; Suneja, Kriti (May 2020). "Hardware Design of Approximate Matrix Multiplier based on FPGA in Verilog". 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS). Madurai, India: IEEE: 496– 498. doi:10.1109/ICICCS48265.2020.9121004. ISBN 978-1-7281-4876- 2. S2CID 219990653

[6]. Valueva , Maria; Value, Georgi; Semyonova, Nataliya; Lyakhov, Pavel; Chervyakov , Nikolay; Kaplun Dmitry; Bogaevskiy , Danil (2019-06-20). "Construction of Residue Number System Using Hardware Efficient Diagonal Function".