PERFORMANCE MEASUREMENT AND OPTIMIZATION OF EXASCALE SYSTEMS

Ashwinkumar Ganesan
Department Of Computer Science and Electrical Engineering (CSEE), UMBC.
Email: gashwin1@umbc.edu

ABSTRACT

In 2009, IBM’s RoadRunner reached a sustained performance of 1 Petaflop, measured using the Linpack Benchmark [1]. With performance at such Petaflop levels, the next target to be achieved in the area of high performance computing is the exascale level where machines will be able to perform, at 1000 Petaflops rate or more. The scale of computing presents a unique set of challenges in terms of performance measurement and application design. In this report, we look at exascale computing, its challenges and a design for performance measurement in exascale computing.

1. INTRODUCTION

Exascale computing is about creating a new generation of supercomputers. These supercomputers have a performance of about 1000 times more than the current generation of supercomputers. The idea is to have computational capability of about a 1000 Petaflops. The actual performance of the application depends on the domain and architecture of the application, i.e. if it has been designed to use supercomputers of exascale size. But 1000 times improvement in performance is a relative and ambiguous term. We can define the improvement in terms of three main attributes [2]:

I. Functional Performance.
II. Physical Performance.
III. Application Performance.

Exascale Computing requires new development for a set of challenges which are related to the hardware and software used as well as designed to attain the high scale of performance. We discuss the challenges in short in the sections ahead. We also discuss the possible software architectures which can be implemented for such systems.

Performance is a critical aspect of exascale systems. Performance Monitoring and Application Optimization is a continuous activity. We look at a possible design for automating performance analysis and management process.

2. ATTRIBUTES TO MEASURE PERFORMANCE

2.1 FUNCTIONAL PERFORMANCE

Function performance metrics of an exascale system are defined across three major attributes:

I. The basic rate of computation i.e. the processing speed of the system. The computational rate takes into consideration the instructions that are executed per second (IPS) and the floating point operations per second (FLOPS).
II. The storage capacity of the system. Storage includes the main memory of the system, the cache memory of system and the persistent storage (disks) used in the system.
III. With better computation and storage, better communication between nodes in an Exascale system is required. This encompasses the network bandwidth, the bisection width of the network, bandwidth between cache and main memory (interconnection bandwidth) and finally the inter-connection speed between various processors in the on a Multicore chip.

2.2 PHYSICAL ATTRIBUTES

The physical attributes of the system include the power consumed, the amount space that is required to host the supercomputer and the cost involved in building the system. The energy consumption is very critical as the large number of cores and processors used will consume a lot of electricity.

2.3 APPLICATION PERFORMANCE

With exascale systems, new application designs are required to take advantage of the computational capacity available. Application designs need to be parallel not only at the logical level, but also at hardware and systems level, where they are able to use multicore machines efficiently.

3. CHALLENGES IN EXASCALE DESIGN

There are many challenges in designing exascale systems. Some of the important challenges are listed below:

I. Power Consumption and Energy Utilization.
II. Memory and Storage.
III. Having a high degree of parallelism and concurrency during application execution.
IV. Resilience and Fault Tolerance.

3.1 POWER CONSUMPTION PROBLEMS

The problem of energy consumption is very important for exascale systems. Technologies need to be built which reduce the total energy consumption per operation. The energy consumption needs to be reduced across the following points:

I. The energy required to send data across the network.
II. The energy required by the memory to store data.
III. The electricity required by the processor on the machine.
IV. The energy required for the secondary storage such as disks and solid state devices (SSD).

As the number of transistors per micron area on the chip has increased, the graph shows the increase in power consumption.

Apart from system itself consuming a lot of energy, the heat generated by the system needs to be removed. Hence a considerable amount of energy is spent in air-conditioning and cooling the system.

3.2 MEMORY AND STORAGE PROBLEMS

Memory problems are present at two levels:

I. To find the memory capacity to store information at exascale levels.
II. To reduce the overall electricity consumed by memory.

Today, the available memory chips are 1GB per chip. To have a Petabyte of memory would require a lot of chips. Data centers designed would have 10 to 100 Petabytes of main memory, making the number of chips to be utilized excessively high. Even if we have main memory of that size, there is a significant bottleneck in bandwidth and intercommunication of data from main memory and other system components [2].

Storage is also big challenge because there is a need for larger data storage units occupying lower volume of space. The speeds at which the secondary storage devices work also needs to be significantly higher. This includes an improvement in communication between secondary storage and main memory and increase read-write speeds of the disks. With larger disks, also comes the problem of creating a file system which can handle such large content. Problems range from managing the data to creating an efficient metadata structure so that the metadata storage is not very high. This problem can become a bigger challenge when concurrency of execution is introduced. For applications using high degree of concurrency in accessing and manipulating data stored in large databases, synchronization and consistency problems can become large.

3.3 CONCURRENCY PROBLEMS

The concurrency problems can be summarized into 3 main parts:

I. The total number of operations that need to be instantiated and executed.
II. The minimum number of threads that need to be running concurrently, so that the required task can be achieved.
III. The number of threads that need to be maintained, to account for some threads executing high latency operations (such as disk operations). This is to maintain the overall concurrency of the system and reduce execution times for the application.
Concurrenc requires a number software design improvements such as:

I. Changes in the program execution stack, to remove unwanted layers.
II. Improvement in the time required to spawn and destroy threads.
III. Compiler need to be improvised to exploit parallelism while compiling the code.
IV. Improvement in Operating Systems design, to make more efficient for memory management and reduce execution times. Operating Systems today are constructed with multiple layers, which can be reduced to make it more efficient.
V. Operating System improvements on branch prediction during execution.
VI. Operating Systems need to have better Job scheduling mechanisms.
VII. Analyzing system security and cost of using secure communication channels.
VIII. Improvements in parallel algorithms.
IX. Synchronization of threads is important. It also increase the number of messages communicated.
X. The size of messages passed will also change the system performance.
XI. Instead of Global Synchronization, analyzing the use of Local Synchronization. Local synchronization can be based on synchronizing threads based on locality of the node or based on where the data block on which it is executing is located.
XII. Algorithms can be use sparse matrix formats to reduce the size of data stored in the main memory and cache.
XIII. Creating checkpoints, is necessary in case of there are faults, but can be a costly operation. Depending upon the size of the current state of the application, a lot of data may have to be stored on the main memory, cache or on the swap space or disk itself. This is based on the function for which checkpoints are used [2].

3.4 RESILIENCE AND FAULT TOLERANCE

Resilience is the property of a system to continue execution of a program, even when there are certain faults in the system such as disk or processor failing.

In case of exascale systems, which have a very large number of components, the number of failures could increase dramatically. For example, a chip with a FIT rate of 1000 fails once every 16 years. This would mean that, we would have a lot of chips failing every 30 seconds in an exascale system. The cost of the fault is also larger because:

I. There is re-computation of data in locations where the faults occur.
II. There is cost incurred in recovery the data lost in case of disk failures.
III. When there is high degree of concurrency, there could be communication losses, in case of section of network were to fail.
IV. Fault tolerance and diagnostics messages to check the health of the system also increase in systems of such a scale.

4. SELF TUNING APPLICATIONS

Self-tuning applications are applications which are aware of their environment and are applications which are able to take corrective measures to improve their throughput. This is can be done by optimizing their internal parameters during program execution.

Apart from performing application tasks, these applications perform 4 additional tasks:

I. Define a baseline for throughput and performance.
II. Continuous measurement of performance.
III. Analyzing the performance.
IV. Taking preventive/corrective actions based on the analysis.

![Figure 3 – OODA Loop for Self Tuning Applications](image)

Measurement of performance can be done by collecting data from the environment. There could be a variety of information such as:

I. CPU Utilization.
II. Memory Utilization.
III. Network / Bandwidth Utilization.
IV. Utilization of the disk.
V. Disk space consumed.
VI. Application specific information i.e. information on database and application servers.

Self-Tuning applications have the advantage of taking preventive measures in case application performance degrades. This helps reduce resource wastage. It can also reduce the time taken to detect faults and thus reduce MTBF.
(mean time between failures). Self-Tuning applications are thus useful at the exascale level.

The applications are adaptive in nature:

I. They have automated mechanisms to enforce Quality of Service parameters to control congestion and flow internal to a node and between nodes [7].
II. Mechanisms for proper communication scheduling and differentiating between short and long range communication.
III. Selecting proper communication primitives based on the current application load and software requirements.

5. DESIGN FOR PERFORMANCE MEASUREMENT (INTELLIGENT PERFORMANCE ANALYTICS CONSOLE – IPAC)

The design is a high level design for a performance measurement application that can benchmark, measure analyze and take actions with regard to system performance.

The information regarding the application is mainly got from 3 sources:

I. The application servers.
II. The database server.
III. The network traffic information.

This information can be available in the form of logs, in the node or machine. The important aspect about using logs is to set a proper log level, so that the information generated is of the right amount. Excess information may take more time to process, which delays the analysis and actions taken by the application to improve the performance. One way to correct this problem is to analyze data iteratively, increasing the log levels, when the information required is not sufficient in the current iteration.

DATA ANALYTICS part of the design is the module which parses the log file content, formats the data, stores the data accordingly and analyzes the data.

PERFORMANCE RULES ENGINE compares the analyzed data against rules established in the system. The main aim is check the content and find if the problem exists in the current system. If a problem exists it stores it. The component thus creates a list of all problems with their likely impact on the system.

ROOT CAUSE ANALYSIS component checks the problem list and defines which problems are important and need to be corrected based on the likely impact of each problem. It also provides the solution to the problem.

The “rules engine” and “root cause analysis” module are can be implementations of problem classifiers such as support vector machines. These classifiers can be used to classify the problem based on location of the problem and the impact. As new problems are encountered, the rules engine updates its set of rules, to classify these problems. The tool also has a way to get feedback from the system and user’s, so that the rules can be changed in case the solution to a problem was not optimal.

6. ENERGY AWARE ALGORITHMS

Energy aware algorithms are algorithms which try to reduce the number of floating point operations or instructions to complete a certain task. They are also aware of the cost of trade-off between operations, such as memory operation and CPU operation, choose accordingly operations which can help save energy.

Energy aware algorithms have to a number of factors into consideration:

I. The cost of each operation in the program.
II. The cost of CPU operations, disk operations and memory related operations.
III. Each of the operations needs to be monitored, and thus there is an added cost collecting energy metrics of the application.

IV. The cost of calculating the energy metrics and providing feedback. The cost or energy consumed for analysis should be as minimal as possible to the actual savings gained by changing the execution method of the application.

V. The increase or decrease in performance of the application. Analyzing the amount degradation in performance if any.

The performance of the algorithm is thus dependent to a high extent on the metrics chosen.

7. CONCLUSIONS

Exascale computing is a new area of active research. Designing adaptive systems for performance measurement, enhancement and energy aware systems can help reduce the cost and scale of the problems in application design.

8. REFERENCES


