Modeling Energy Consumption of a Parallel Application on a Computational Grid

Nandini S[1], Chithra Apoorva D A[2], Mala B M[3]
[1,2,3] Assistant Professor, Dept of CS & E, GITAM School of Technology, Bangalore.

Abstract : Energy, non-renewable resource has at-most global importance. Conserving this energy in computational grids especially in large data centers has been a hot topic of discussion since many years. A small amount of reduction in energy consumption in a single node would add up resulting in a significant amount energy savings in the data centers. In this paper the energy model is derived based on various parameters that are obtained using PAPI and Perf, a Linux profiler. The work also concentrate on scaling Central Processing Unit frequency and obtain results for various workload. Molecular dynamics a systolic approach is the program that is used. The program is run at different workloads and at different CPU frequencies. Based on that, energy equation is derived and graphical representation of the results are shown for different frequency ranges and workloads. The result can be utilized to identify the CPU frequency for running the applications which can provide better performance in acceptable amount of time with reduced energy consumption.

Key terms: Grid, PAPI, Perf, Molecular Dynamics

1INTRODUCTION

Measuring performance of any of the programs or applications is very much important because the system resources are very expensive. Especially while running programs in grid environment it must be given higher importance. Higher the usage of Central processing Unit and other resources of computers connected might result in higher power dissipation and consume lot of energy. Higher energy use is not just waste of expensive energy but it will effect environment. Higher the release of carbons to the environment will result in greenhouse effect. So, it is very much important to analyze the performance of any applications or programs especially in Grid or cloud. So, the programs and applications must be optimized.

Grid computing is a collection of multiple computer and computer resources spread across different geographical locations, must be connected by a computer network and all working towards a goal simultaneously. The systems can be heterogeneous with different operating systems, different RAM (Random Access Memory) capacity and different cache, that is systems with different hardware and software configurations. But all must be connected by a computer network.

The grid can be used for solving intensive scientific computations, mathematical and analytical problems. It can be used for solving computations that might require very less or no communication between the running processes.

Optimization: Optimization of programs helps in reducing the time and resources utilized in executing the program. To optimize a program, that is to find the points in the program where large number of CPU cycles or any other system resources are being consumed is very much important. This can be done with the help of Profilers.

PAPI (Performance Application Programming Interface) provides operating system independent access to CPU hardware performance counters. It has 2 layers a machine dependent layer and a machine independent layer which will access the hardware counters of the system and provide details such as number of instructions executed, number of clock cycles, cache hits, cache misses etc.

2 RELATED WORK

In the previous work [1], “Measuring Energy and Power with PAPI ” by, Vincent M. Weaver, Matt Johnson, Kiran Kasichayanula, James Ralph, Piotr Luszczek, Dan Terpstra, and Shirley Moore Innovative Computing Laboratory University of Tennessee. In this paper author explains how PAPI can be used to measure energy and power values for CPUs and GPUs internally and externally with external power meters. It also explains usage of available new APIs of PAPI can be used to obtain any particular event of interest to the user. They also explained how PAPI supports reading measurement data of running processes. The author explains about the external power measurement devices like Watt’s Up Pro meter and PowerMon2. Watt’s Up Pro Meter is an external measurement device the system can plug into. It provides power measurement and time using which energy can be calculated. PAPI is used to provide measurements every second irrespective of whether it is requested or not. PAPI component for PowerMon2 is not available yet.

The author also explains about internal measurement of power where energy and power consumption are measured internally. Here the author discuss about Intel RAPL (Running Average Power Limit), AMD Application Power management and NVIDIA Management library. RAPL’s internal circuitry can estimate current energy usage based on a model driven by hardware counters, temperature, and leakage models. The results of this model are available to the user via a model specific register (MSR).

Accessing MSRs requires ring-0 access to the hardware; which can be done only by operating system kernel. So accessing RAPL values require a kernel driver, which is not available with Linux. To solve this the MSR driver is enabled and given proper read-only permissions then PAPI can access these registers directly without kernel support. AMD Application power management, AMD family 15h processors report power in Watts via the processor power in TDP. PAPI support for this is yet to be available. NVIDIA GPU's report power via the NVIDIA Management Library. The nvmlDeviceGetPowerUsage() routine exports the current power.

In [2], "Collecting Performance Data with PAPI-C" by Dan Terpstra, Heike Jagode, Haihang You, Jack Dongarra, The University of Tennessee. In paper[2], the author explains about Component PAPI or PAPI-C, which provides a way to collect multiple sources of performance data simultaneously via a common software interface. The author describes about preset events-a single event native to a given CPU, native events of PAPI and LM-Sensors Component, this component allows PAPI to access computer health monitoring sensors that are provided by lm_sensors library. The author demonstrates how PAPI-C was used to measure CPU fan speed monitoring, CPU temperature monitoring on an Intel Nehalem (core i7) machine using the LM-SENSORS component.

In [3,4], "Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events" by W. Lloyd Bircher and Lizy K. John, Laboratory for Computer Architecture, Department of Electrical & Computer Engineering, University of Texas at Austin. The researchers make use of microprocessor performance counters for online measurement of complete system power consumption. They make use of actual system running scientific and commercial workloads to develop and validate power models for memory, chipset, Input/Output, disk and microprocessor. Memory power model is done using the number of read/write cycles. These are estimated by counting memory bus accesses by processor and other events. Disk power model is done by accurately modelling the amount of time spent moving the disk read/write head and the speed of rotation. Input/output and chipset power model is done by estimating dynamic power in the input/output and chipset subsystems through the number of interrupts, DMA(Direct Memory Access) and un-cacheable accesses. Author makes use of on-chip performance counters reading through CPU (Central Processing Unit) register accesses. They make use of dynamic adaptation which involves reconfiguring hardware dynamically to match software demands. By this high performance and low power consumption can be achieved. To do this phase detection needs to be done that is identifying performance insensitive phases of program execution. Their subsystem consists of resistors connected in series with the power source, the voltage drop across resistor is used to measure power consumed by the subsystem since voltage drop across resistor is directly proportional to the power consumed. Performance events in the processor are recorded at rate of once per second.

For experimenting eleven workload were considered which involve eight workloads form the SPEC CPU 2000 benchmark. SPEC workloads include computationally intensive scientific applications intended to stress the CPU and memory subsystems. Events such as cycles, halted cycles, fetched uops, level 3 cache misses, TLB misses, DMA accesses, processor memory bus transactions, un-cacheable accesses and interrupts were noted down and the power consumed in Watts were tabulated for all the eleven workloads namely idle, gcc, mcf, vortex, art, lucas, mesa, mgrid, wupwise, dbt-2, SPECjbb and DiskLoad. The below equations were derived.

SMP Processor power model derived using gcc workload is given by:

$$\sum_{i=1}^{NumCPUs} 9.25 + (35.7 - 9.25) \times PercentActive_i + 4.31 \times \frac{FetchedUops_i}{Cycle}$$

Cache miss memory power model derived using mesa workload is given by:

$$\sum_{i=1}^{NumCPUs} 28 + \frac{L3LoadMisses_i}{Cycle} \times 3.43 + \frac{L3LoadMisses_i^{\,2}}{Cycle} \times 7.66$$

Input/output power model using synthetic workload is given by:

$$\sum_{i=1}^{NumCPUs} 32.7 + \frac{Interrupt_i}{Cycle} \times 108 \cdot 10^6 - \frac{Interrupt_i^{\,2}}{Cycle} \times 1.12 \cdot 10^9$$

3 PROPOSED WORK

The below figure 3.1 depicts the flow chart of the proposed work

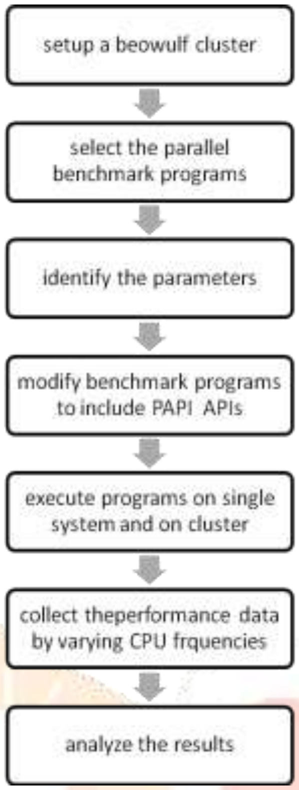

setup a beowulf cluster
select the parallel benchmark programs
identify the parameters
modify benchmark programs to include PAPI APIs
execute programs on single system and on cluster
collect theperformance data by varying CPU frquencies
analyze the results

Figure 3.1 Flowchart of proposed work

A. Setting a Beowulf cluster: Beowulf is a method of creating clusters. It is not a software but a way of building a supercomputer. We can make use of any computer hardware resources that are available like memory disks, Central Processing Units, Monitors that are not being used a supercomputer can be built even with a single mouse and keyboard. This way by using waste computer resources a super computer can be built.

B. Selecting benchmark program: By using benchmarks we can compare the performance of various subsystems across multiple chip or system architecture. The benchmark selected is molecular dynamics systolic process.
Molecular dynamics a cpp program, is a computation that involves following the particles path which exert a distant dependent force on each other. If particles meet, they pass through other. In the molecular dynamics program, the temperature and energy consumed by systems are calculated by varying the number of particles.

C. Measuring energy and power: Energy and power are the most important resources. To compute energy and power we are using number of instruction, number of cpu-cycles executed, L1 cache misses, L2 cache misses and execution time parameters of program execution. PAPI and perf are used to obtain the above parameters of program execution. PAPI provides APIs to compute these parameters and perf provide these values by directly reading the LINUX events.
PAPI comes with 2 layers, a machine dependent layer and a machine independent layer. By compiling and statically linking independent layer with hardware specific layer, PAPI library instance will be produced for specific operating system and hardware architecture. PAPI portable layer calls the substrate, the internal PAPI layer which handles machine dependent specifics of accessing counters. Substrate uses all the available methods that are appropriate to facilitate counter access. When particular hardware event exceeds specific threshold, it calls user-defined handlers. Using PAPI almost all hardware counters can be measured by using the APIs.
The APIs provided by PAPI that are utilized in this paper are:

- **PAPI_TOT_INS**: Gives the number of instructions executed or the number of instructions completed. Number of instructions a computer can execute depend on the instruction type, order of execution and the presence of branch instructions.
- **PAPI_TOT_CYC:** Gives the number of CPU cycles used in executing the program. Every instruction goes 5 phases during execution. Each instruction undergoes a fetch, decode, execute, memory access and write back cycle. 1 clock cycle is the time between 2 pulses of an oscillation. This can be used to determine speed of a computer processor.

- **PAPI_L1_DCM**: Gives number of cache hits of L1 level. L1 cache is usually built into microprocessor chip of the processor itself. It will be smaller in size.
- **PAPI_L2_DCM**: Gives number of cache hits of L2 level. L2 cache is on a separate chip and it will be of larger size than L1
- **PAPI_get_real_cyc( ):** Gives the real time counter value in clock cycles. It returns the total amount of real time that is passed since some arbitrary starting point. This time is returned in clock cycles.
- **PAPI_get_real_usec( ):**Gives the real time counter value in microseconds. It is same as PAPI_get_real_cyc but returns time in microseconds
- **PAPI_library_init( ):**Initializes the PAPI library. It needs to be called mandatorily before using any of the low level PAPI functions.
- **PAPI_num_counters( ):**Which returns the hardware counters that are available on the system.
- **PAPI_num_events( ):** Returns the number of events that are mentioned in the eventset.
- **PAPI_start_counters( ):** High level API which starts counting hardware events. It will start counting events in event array whenever it is called for.
- **PAPI_stop_counters( ):** High level API which stops counting hardware events. It will stop counting events in event array whenever it is called for.

The below figure 3.2 illustrates the structure of PAPI

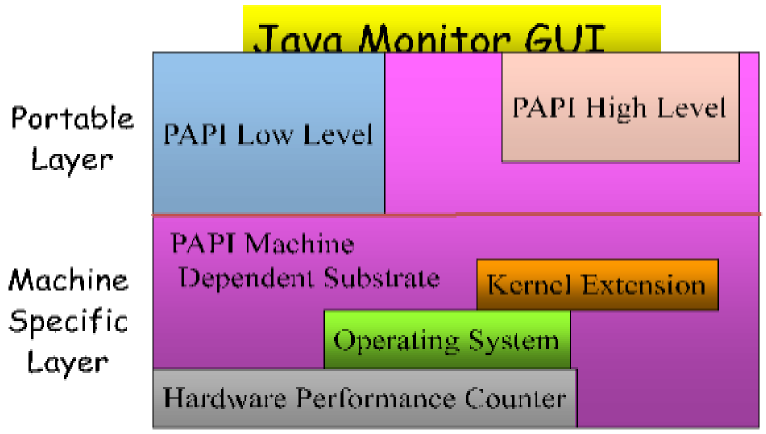


Figure 3.2: PAPI Structure

PERF is Linux profiler, which is also called as Performance counters for Linux. The events reported by perf are called perf events. While PAPI provides only hardware events, Perf can provide software, hardware, hardware PMU events and trace back events. Perf tool is available in linux-tools-common package. It can be installed by executing command **sudo apt-get install linux-tools-version.** The version should match the kernel . All the events that can be measured PERF are provided by **perf list** command. Option *u* is used along with perf stat to obtain program specific performance parameters. The events that are used in the molecular dynamics program are:

- CPU-cycles
- Instructions
- L1-dcache-misses
- LLC

CPU-cycles is used to obtain total number of cycles used for program execution. Instruction gives the total number of instructions. Utilizing CPU-cycles and instructions, cycles per instruction can be calculated. But perf provides number of instructions by default. L1 cache and L2 cache values can be used to show that increase in L1 and L2 cache misses will increase the number of cpu-cycles.L1-dcache-misses provides the data cache misses and LLC provides the last level cache misses. Execution time is provided by perf automatically.
Using the CPU-cycles and the time elapsed during program execution, the energy consumption of the program can be obtained. Number of CPU-cycles is directly proportional to the amount of energy consumed.

4 SYSTEM MODEL

The system model shows the number of processor elements in a cluster. For a system of *n* clusters with *m* nodes in each cluster, the processing elements of a cluster is given by:

$$C = \sum_{i=1}^{n} \sum_{j=1}^{m} Pij \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \text{Eqn 4.1}$$

Where,

n=number of clusters
m=number of nodes in each cluster and
p = number of cores per node

The below figure 4.1 shows the system model

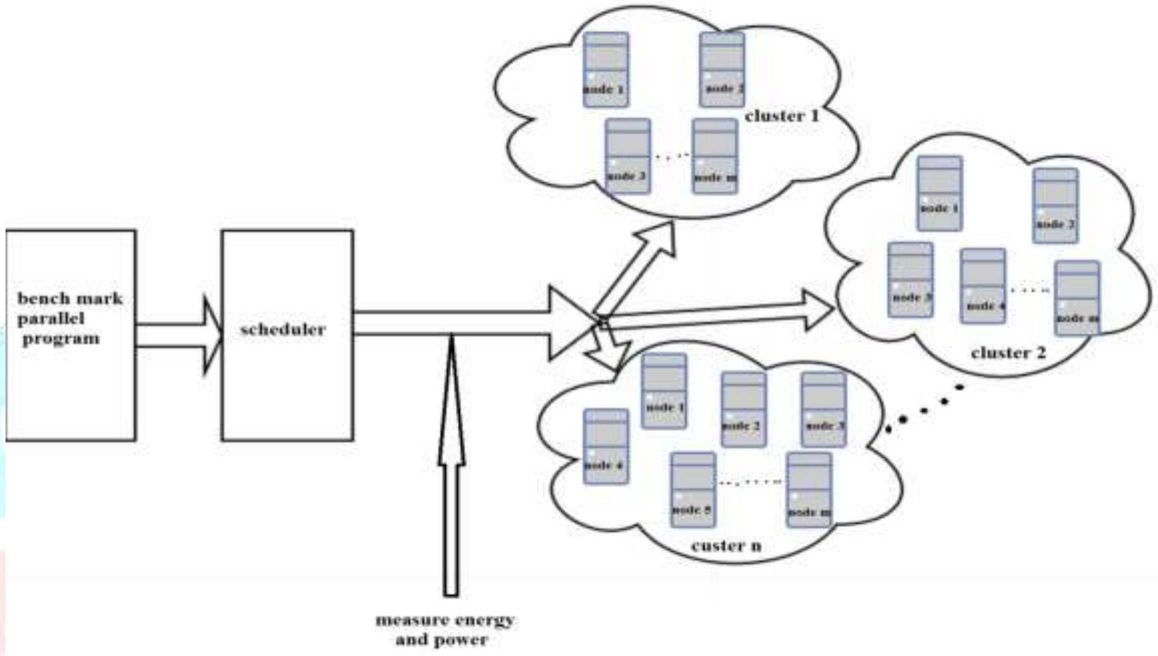


Figure 4.1 (a) System Model

4.1 Energy Model

By performing the experiment it is observed that the energy consumption is directly inversely proportional to the number of clock cycles. Based on this Energy equation can be derived using power, clock cycles utilized during execution and the base speed of the processor.
Power equation is given by:

$$P = Dw * [\left(\frac{fmin}{fmax}\right) * \left(\frac{Vmin}{Vmax}\right)]^2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Eqn 4.2}$$

Where,

P = Power consumed
Dw = Default voltage =64Watts,
fmin = minimum frequency = 1.6GHz
fmax = maximum frequency = 2.6GHz
Vmin = minimum voltage = 0.85V and
Vmax = maximum voltage = 1.36V

Energy equation is given by:

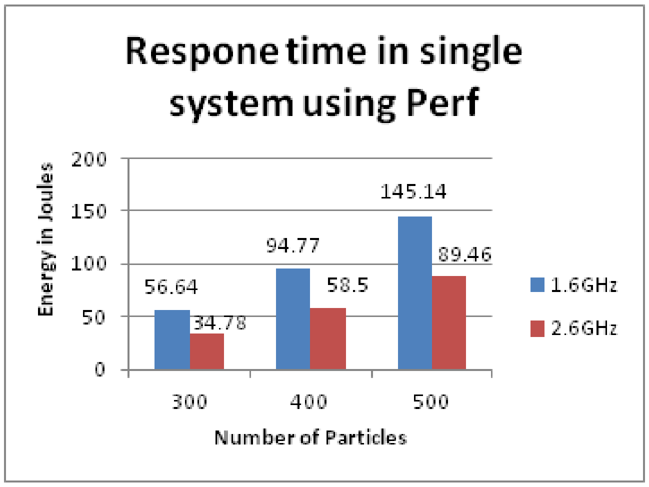
$$E = P * \Delta t/(speed\ of\ processor)\ \ldots\ldots\ldots\ldots\ldots\ldots \text{Eqn 4.3}$$

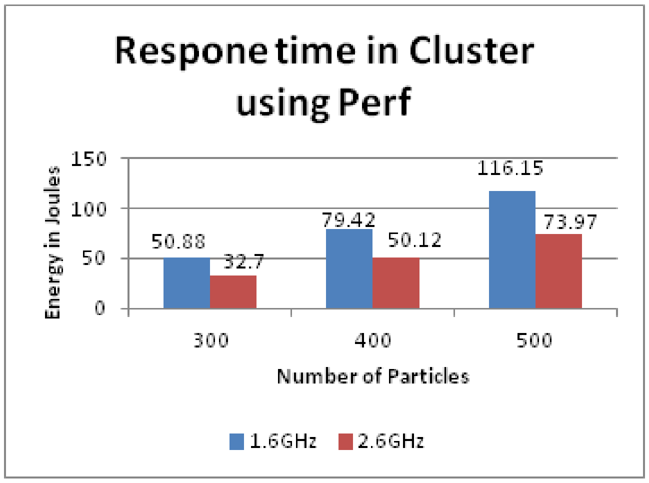
Where,

E = Energy consumed in joules
P = power obtained by previous equation and
Δt = number of clock cycles

5 PERFORMANCE EVALUATION

The results obtained using PAPI and Perf for 1.6GHz and 2.6GHz are analyzed for 3 different workloads at 300, 400 and 500 particles. The graphs are drawn for response time and energy consumed against different workloads

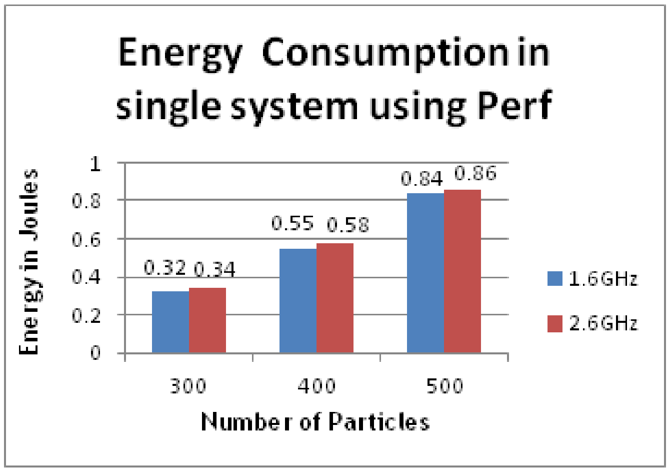


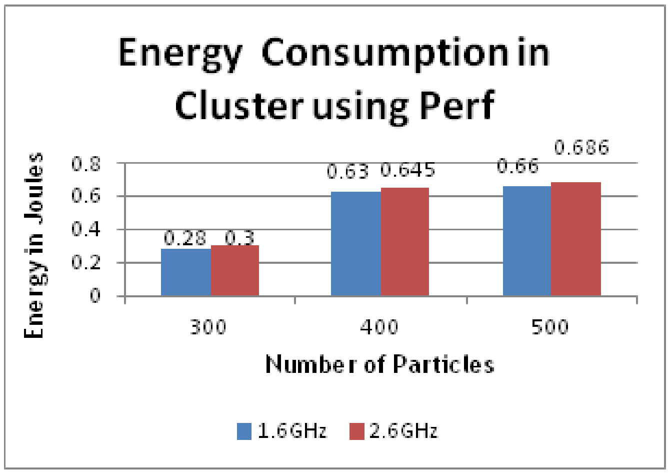


(a) **(b)**

Figure 5.1(a):Graph representing response time of different workloads in a single system using perf. (b) Graph representing response time of different workloads in cluster using perf.

The graph 5.1(a) shows the response time of different workloads in a single system using perf with 1.6 GHz and 2.6 GHz . With 1.6 GHz the response time is high when compared to 2.6 GHz. As the number of particles increases the response time also increases in a single system using perf. But the response time is low in cluster with respect to perf. Hence using perf the response time is reduced in cluster.

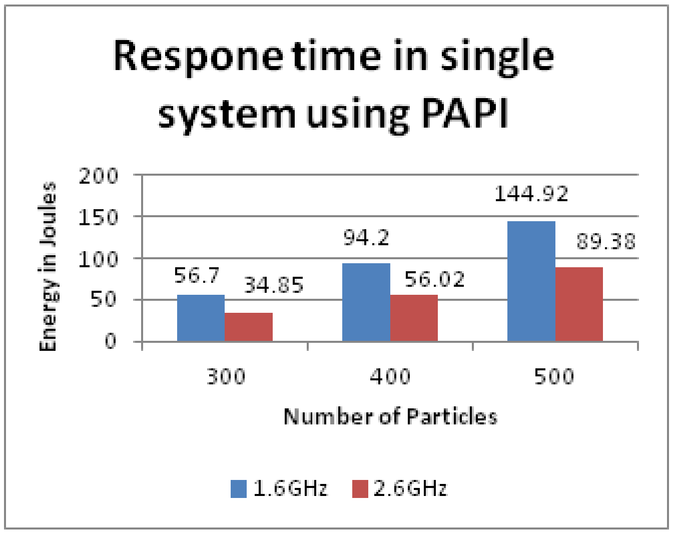


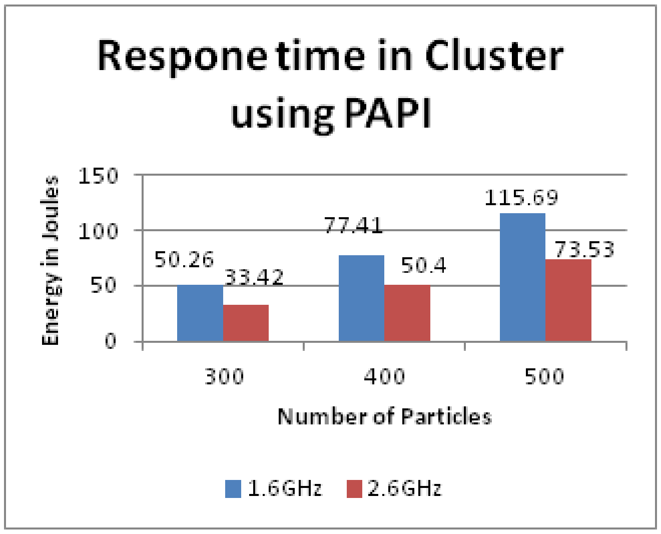


(a) (b)

Figure 5.2 (a): Graph 3 representing energy consumption of different workloads in a single system using perf. (b) Graph representing energy consumption of different workloads in a cluster using perf.

The graph 5.2(a) shows the energy consumption in a single system using perf with different number of particles. Here, the energy consumption does not vary much as the number of particles increase with different CPUs of 1.6 GHz and 2.6 GHz. In graph 5.2(b), the energy consumption in cluster with perf remains almost same with two different processors as the number of particles increases. But the energy consumption in reduced when perf is used in cluster which is considered to be an important factor in clusters



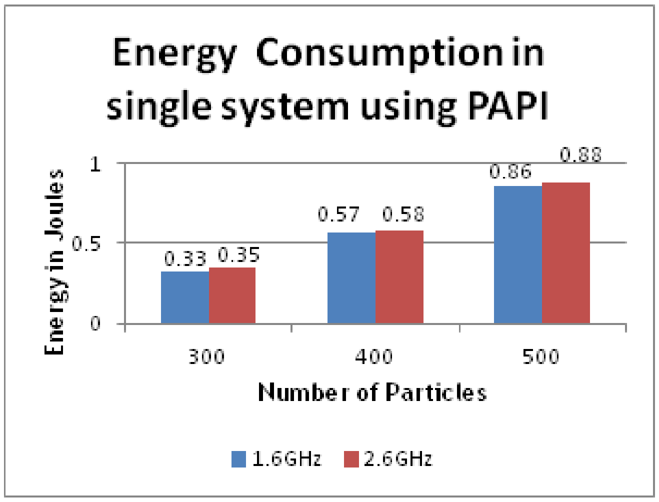


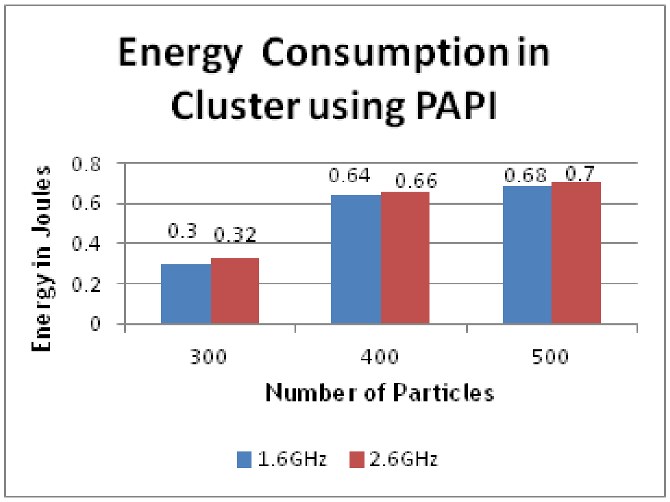
(a) (b)

Figure 5.3(a): Graph representing response time of different workloads in a single system using PAPI. (b) Graph representing response time of different workloads in a cluster using PAPI.

The graph 5.3(a) shows the response time of different workloads in a single system using PAPI with 1.6 GHz and 2.6 GHz . With 1.6 GHz the response time is high when compared to 2.6 GHz. As the number of particles increases the response time also increases in a single system using PAPI. But the response time is low in cluster with respect to PAPI. Hence using PAPI the response time is reduced in cluster. Using PAPI in cluster the response time is very much reduced when compared to perf. as well as the energy consumption is also reduced

The graph 5.4(a) shows the energy consumption in a single system using PAPI with different number of particles. Here, the energy consumption does not vary much as the number of particles increase with different CPUs of 1.6 GHz and 2.6 GHz. In graph 5.4(b), the energy consumption in cluster with PAPI remains almost same with two different processors as the number of particles increases. But the energy consumption in reduced when PAPI is used in cluster when compared with single system using PAPI which is considered to be an important factor in clusters





(a) (b)

Figure 5.4(a): Graph representing energy consumption of different workloads in a single system using PAPI.(b) Graph representing energy consumption of different workloads in a cluster using PAPI.

5. CONCLUSION AND FUTURE WORK

The energy consumption in cluster in an important factor and hence it is analyzed using perf and PAPI. It can be concluded that using perf and PAPI in clusters can reduce the response time and energy consumption.
The energy is modeled considering the default wattage, minimum frequency, maximum frequency, minimum voltage, maximum voltage, baseline frequency of the system and the number of clock cycles. Energy model is derived by varying CPU frequency and executing the program at different workloads.
This work can also be extended for other more number of particles and can be implemented with different program as in this paper it is concentrated on molecular dynamics.

The above work can be carried out on multiple nodes cluster. In this paper we have used 2 nodes in a cluster. By carrying out the experiment in more nodes the values can be much accurate, experiment can also be carried out at multiple frequency levels. To obtain the optimal frequency for operation. In the energy model temperature of the system is not considered. Increase in temperature will increase energy consumption of system. So, in the future temperature can also be used as a parameter to model energy.

REFERENCES

[1] “Measuring Energy and Power with PAPI ” by, Vincent M. Weaver, Matt Johnson, Kiran Kasichayanula, James Ralph, Piotr Luszczek, Dan Terpstra, and Shirley Moore Innovative Computing Laboratory University of Tennessee
[2] “Collecting Performance Data with PAPI-C” by Dan Terpstra, Heike Jagode, Haihang You, Jack Dongarra, The University of Tennessee
[3] “Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events” by W. Lloyd Bircher and Lizy K. John, Laboratory for Computer Architecture, Department of Electrical & Computer Engineering, University of Texas at Austin.
[4] “Power-Aware Predictive Models of Hybrid (MPI/OpenMP) Scientific Applications on Multicore Systems” by Charles Lively · Xingfu Wu · Valerie Taylor · Shirley Moore · Hung-Ching Chang · Chun-Yi Su · Kirk Cameron
[5]] "Energy and performance characteristics of different parallel implementations of scientific applications on multicore systems” by Charles Lively, Xingfu Wu, Valerie Taylor, Shirley Moore, Hung-Ching Chang and kirk Cameron
[6] “An energy-efficient scheduling algorithm using dynamic voltage scaling for parallel applications on clusters” by Xiaojun Ruan, Xiao Qin, Ziliang Zong, Kiranmai Bellam, Mais Nijim, Department of Computer Science and software Engineering, Auburn University, Auburn and Department of Computer Science, University of Southern Mississippi, Hattiesburg
[7] “Profile-based Dynamic Voltage and Frequency Scaling for a multiple clock Domain Microprocessor” by Grigorios Maglis, Michael L Scott, Greg Semeraro, David H Albonesi and Steven Dropsho, Department of Computer Science, Department of Electrical and Computer Engineering, University of Rchester, Rochester,NY.
[8] "The new Linux ‘perf’ tools” by Arnaldo Carvalho de Melo, Red Hat Inc, Brazil
[9] Perf Tool: Performance Analysis Tool for Linux
[10] PAPI: Performance Application Programming Interface, Walt Mankowski

[11] "Energy consumption model over parallel programs implemented on multicore architectures" by Ricardo Isidro-Ramfrez, Amilcar Meneses Viveros Erika Hernandez Rubio, SEPI-ESCOM Mexico, D.F.
[12] "Energy Consumption on Heterogeneous Computing Platforms" by Savina Bansal, Kaushal Bansal, R K Bansal, Department of Electronics and communication engineering, Punjab Tech University Campus, India
[13] "A Parallel Model and Runtime System for Approximation-aware Heterogeneous Computing" by Ioannis Parnassos, Nikolaos Bellas, Nikolaos Katsaros, Nikolaos Patsiatzis, Athanasios Gkaras, Konstantinos Kanellis, Christos D. Antonopoulos, Michalis Spyrou and Manolis Maroudas University of Thessaly Volos, Greece
[14] "A Formal Approach to the Mapping of tasks on an Heterogeneous Multicore, Energy-aware architecture", by Emilien Kofman, Robert de Simone, Inria
[15] "An efficient dynamic energy-aware application mapping algorithm for multicore processors", by Thomas Canhao Xu, Ville Leppanen, Department of Information Technology, University of Turky, Finland