Power and Energy-Aware Processor Scheduling

Luigi Brochard, Francois Thomas
IBM Systems and Technology Group
Montpelier, France
{luigi.brochard, ft}@france.ibm.com

Raj Panda, Don DeSota, Robert H. Bell, Jr.
IBM Systems and Technology Group
Austin, Texas
{panda, desotad, robbell}@us.ibm.com

ABSTRACT

Power consumption is a critical consideration in high computing systems. We propose a novel job scheduler that optimizes power and energy consumed by clusters when running parallel benchmarks with minimal impact on performance. We construct accurate models for estimating power consumption. These models are based on measurements of power consumption on benchmarks with different characteristics and on systems with processors using different micro-architectures. We show the power estimation models achieve less than 2% error versus actual measurements. We show a job scheduler can be enhanced to make it "power-aware" and to optimize power consumption of jobs with similar performance characteristics. The enhanced scheduler can estimate the power consumed by a particular job using the power estimation model, configure the nodes in the cluster via suitably adjusting processor frequency on each of the nodes to maximize performance, minimize power, or minimize energy with a predictable impact on power, energy and performance.

Categories and Subject Descriptors

B.8.2 [**Hardware**]: Performance Analysis and Design Aids

General Terms

Algorithms, Measurement, Performance, Design.

Keywords

Power Analysis, Performance, Power Estimation, Power Optimization.

1. INTRODUCTION

Recently, power consumption has become a serious concern to managers of HPC data centers due to the rising cost of power and cooling. While data center managers are interested in optimal management of server system power allocation to minimize total operational cost of the data center, a typical HPC user is interested in the best turnaround time or for overall throughput of his job. Realizing the importance of power management, hardware vendors are building more and more dynamic power management capabilities into microprocessors and server systems as well as providing software tools to obtain and view the power consumption data from the server systems. Some of the available tools can also be used to set limits on the power delivered to the server systems and thus help data center managers in the management of power and cooling costs. However, these software tools are not targeted to parallel applications and do not predict the impact of processor frequency scaling on total energy. For example, cases are described in this paper in which decreasing the processor frequency increases the energy consumed by the application while power consumption is decreasing. Since current schedulers cannot predict the power dissipation of their applications, optimized power and energy management of the cluster nodes is difficult.

Recognizing the HPC application community's need, we study performance and power consumption on a selection of HPC applications. Our first objective is to experimentally obtain generalized power-performance correlations for HPC applications that can be used to estimate the power consumption and energy of an application, on any platform, and at any frequency. Then, a scheduler is implemented to obtain an executing job's power dissipation and make use of the derived power-performance correlations to optimize the power dissipation and energy of the job executing on the HPC cluster. We use IBM POWER6 and Intel Harpertown and Nehalem server results and analysis to carry out these tasks.

Floyd et al. [2] and McCreary et al. [3] describe the system power management support in the POWER6 processor. Techniques such as core throttling and power and temperature monitoring capabilities are discussed. Allarey et al. [5] describe idle and multicore dynamic power reduction features in Intel's 65nm cores, and they introduce a deep power-down idle state and power-performance tradeoffs for single threads, as well as enhanced sleep states.

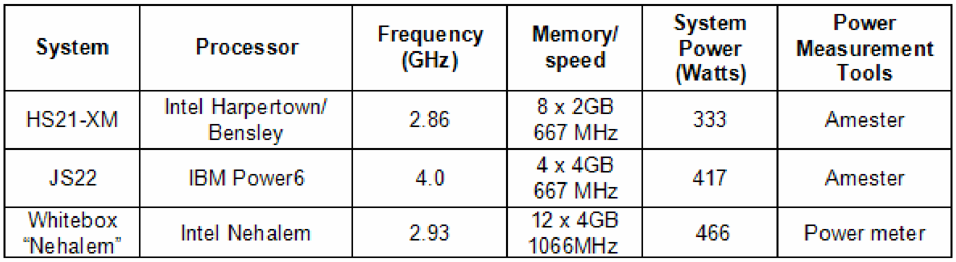
Rajamani et al. [6] propose real-time power and performance prediction capabilities that can be used for dynamic control of system resources such as DVFS and clock throttling to improve power-performance. They extend prior work related to average power prediction to predicting instantaneous processor power to enable applications such as operating system scheduling. Lee et al. [7] dynamically predict performance and power using regression models and apply them to controlling DVFS for program regions of 100M instructions.

Our work is distinguished from prior work by its focus on power-performance in large, high-performance systems of many clustered nodes. The possible dynamic variations of node power, application power and energy in such a system, including a maximal system power constraint, require accurate, dynamic power prediction on an application basis and unique algorithms for predicting and controlling power and energy system-wide, as proposed here.

The rest of the paper is organized as follows. Section 2 gives a brief description of the machines used in the power-performance experiments performed for this paper. Section 3 gives a

ICPE'11, Month 14-16, 2011, Karlsruhe, Germany.

Table 1: Benchmark Systems



System	Processor	Frequency (GHz)	Memory/ speed	System Power (Watts)	Power Measurement Tools
HS21-XM	Intel Harpertown/ Bensley	2.86	8 x 2GB 667 MHz	333	Amester
JS22	IBM Power6	4.0	4 x 4GB 667 MHz	417	Amester
Whitebox "Nehalem"	Intel Nehalem	2.93	12 x 4GB 1066MHz	466	Power meter

description of the performance data gathering process for both Power6 and Core 2 micro-architectures and the derived metrics used in the models. Section 4 presents the tools used to measure power consumption. Section 5 presents the applications used for this study. Section 6 presents performance and power measurements gathered on the various platforms and the model used for power and performance projections. Section 7 presents the impact of frequency scaling on power and energy for the benchmarks on the various platforms. Section 8 uses the data from previous sections and proposes the power and energy aware scheduling methodology and policies and quantifies the power-performance impact of the methodology. Section 9 presents the conclusions.

2. Experimental Systems

The systems used in our experiments are an IBM JS22 blade, an IBM HS21-XM Harpertown blade, and a Nehalem "white box" described below.

Table 1 shows configuration details for the systems used in the current work. JS22 is a blade that has two POWER6 modules running at 4.0 GHz. Each module has two cores capable of running in either single threaded (ST) mode or multi-threaded (SMT) mode, with two threads per core. The POWER6 chip is a high frequency, in-order superscalar processor with 8MB of L2 cache, an L3 controller, an on-board fabric with controller, and integrated memory controller. In the blade, each POWER6 chip is a single-chip module without an L3, one memory controller, with only two channels attached to the memory DIMMs through buffer chips [1]. The POWER6 core pipeline has two binary floating-point units each capable of two floating-point operations per cycle, for a total of eight sustained floating-point operations per cycle per chip.

A number of JS22 blades can be inserted into the IBM BladeCenter-H chassis, which modularizes the blade power supplies, switch bays, and point-to-point wiring through the backplane [4]. Each blade contains four angled DIMM slots of 8GB DDR2-533 DIMMs or 1GB, 2GB, or 4GB DDR2-667 DIMMs, for up to 32GB of memory, supporting ECC, chip-kill and redundant bit steering. The blade also integrates the peripheral chips. Linux RHEL 4.6 and AIX 53L operating systems are supported. The BladeCenter integrates a management module with support for tools such as IBM Director, Power Executive, and the Active Energy Manager (AEM).

The IBM HS21-XM blade (characteristics listed in Table 1) is also deployed in the IBM BladeCenter-H chassis, and is managed by the same tools as the JS22. It has two sockets of Intel Harpertown Core 2 quad-core processors running at 2.8 GHz. The Harpertown processor cores also execute 4 floating point operations per cycle giving a total 16 floping point operations per cyle per chip. The Harpertown processor is an MCM consisting of two dual core processor chips. The two cores share a common 6MB L2. The MCMs connect to a central memory contoller chip via a "front side bus" running at 1333 Mhz.

The Nehalem system is a 2U "whitebox" with 2 sockets, each socket with one quad-core Nehalem processor running at 2.93GHz and 12 direct-attach 2GB DIMMs running at 1066MHz. It has private L1 and L2 caches per core and a shared 8MB L3. Nehalem also features an integrated memory controller and QPI interconnect.

3. Applications

For an effective analysis of power-performance, a set of floating point benchmarks was chosen that stress either the processor or the memory in the system, or both. A subset of 8 out of a possible 17 of the SPEC CPU2006 benchmarks was chosen in order to speed up the data collection and analysis tasks and to represent different benchmarks that are important to HPC as well as for their different performance characteristics in terms of CPI (cycles per instruction) and memory bandwidth. These are not true parallel applications but measurements show little difference in power between the parallel versions of these workloads and the SPEC FP counterpart. Table 2 shows the selected applications.

Table 2: List of applications and HPC areas

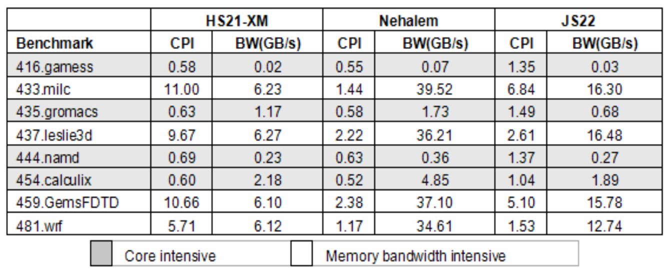
Benchmark	Area
416.gamess	Quantum Chemistry
433.milc	Physics
435.gromacs	Molecular Dynamics
437.leslie3d	Fluid Dynamics
444.namd	Molecular Dynamics
454.calculix	Structural Analysis
459.GemsFDTD	Electromagnetics
481.wrf	Weather Forecasting

4. Performance Metrics

To carry out our experiments, performance counter data from all three machines was collected.

For gathering the counter data, we used hpmcount tool on JS22, oprofile on HS21-XM and perfmon on the Nehalem platforms. Performance metrics like CPI (cycles per instruction) and memory bandwidth were computed for each of the SPEC benchmarks based on the hardware counter data collected using these tools. We ran the SPEC benchmarks in the throughput mode to assess the capability of each system. A number of copies of each of the benchmarks in Table 1 were on each platform for gathering the hardware counter data. On JS22 and the Nehalem platforms, we used the systems in SMT (simultaneous multi-thread) mode which implies that we would run twice the number of copies as the number of cores in the system. In other words, we ran one copy of the benchmark for each logical CPU in the system. Based on the elapse time for the throughput benchmarks to complete on a system, one can also compute the throughput performance called "rate" according to SPEC benchmark rules. In Table 3, CPI and memory bandwidth metrics are shown for each of the benchmarks on each of the three systems.

Table 3: Applications and their performance characteristics on the different systems



	HS21-XM		Nehalem		JS22	
Benchmark	CPI	BW(GB/s)	CPI	BW(GB/s)	CPI	BW(GB/s)
416.gamess	0.58	0.02	0.55	0.07	1.35	0.03
433.milc	11.00	6.23	1.44	39.52	6.84	16.30
435.gromacs	0.63	1.17	0.58	1.73	1.49	0.68
437.leslie3d	9.67	6.27	2.22	36.21	2.61	16.48
444.namd	0.69	0.23	0.63	0.36	1.37	0.27
454.calculix	0.60	2.18	0.52	4.85	1.04	1.89
459.GemsFDTD	10.66	6.10	2.38	37.10	5.10	15.78
481.wrf	5.71	6.12	1.17	34.61	1.53	12.74

▨ Core intensive □ Memory bandwidth intensive

Significant differences between these applications are apparent. Life Science applications like 416.games, 435.gromacs 444.namd are very core intensive and have very little memory bandwidth. 454.calculix which is a structural analysis application is also core dominant. The remaining applications 433.milc, 437.leslie3D, 459.GemsFDTD and 481.wrf have high memory bandwidth requirements.

5. Tools for power measurement

Two system management tools were used to collect the power data used in our experiments. Amester is a tool that is internal to IBM that we have used to measure power at a component level in the blade server while AEM (Active Energy Manager) is a commercially available tool that can be used to measure power at the server level as well as power at chassis level.

5.1 Amester

The Autonomic Management of Energy (AME) project was started at the IBM Research Lab in Austin in 2004 with the goal of controlling server or blade power-performance to within a specified power and temperature budget [2]. JS22 and HS21-XM blades were provided with on-board power-measurement circuits and firmware additions to monitor the circuit outputs. An AME circuit places a very low impedance resistor in series with a power rail. Circuits are placed on the various rails feeding the voltage regular modules that power the system. The on-blade temperature and power management device (TPMD) then converts the voltage drop on each resistor to digital, which allows it to project the current and power at the rails. The Blade Center management module can then interface digitally to the TPMD through the service processor to read the power and other information coming from the rails and control the behavior of the POWER6 through actuators positioned on the die [2].

The rails that are accessible readily are denoted in this paper in the following way:

Core Power (or V_{dd}): Power to the on-chip cores, internal fabric, memory controller, L2 cache controller, and other internal chip logic except for the L2 cache arrays (V_{cs}), I/O pins (V_{io}), and standby logic (V_{sb}).

Total Power (or 12V Power Supply): Power to the entire blade, including the POWER6 chip, L2 cache arrays, I/O pins, standby circuitry, blade service processor, and TPMD.

DIMM Power (or V_{dram}): Power to the memory DRAM and DIMM subsystem.

In the following paragraphs, one additional category of power dissipation, Other, consists of the blade components, other than the chips and memory subsystem, that have relatively static power dissipation (except for the L2 cache arrays, I/Os, and standby logic). This is computed as:

Other Power = Total Power – 2*Core Power – DIMM Power

The power dissipation in the L2 cache arrays and I/Os can vary with the benchmark, but the swing in power is small relative to the overall power of the POWER6 chips and the rest of the system, usually less than 6W, or 2% of the Total Power.

Amester is a companion API tool [2] that provides several interface capabilities: 1) robust network connectivity, 2) timely collection of data, 3) a command line interface to access AME firmware commands, and 4) A GUI to provide visual feedback of the AME firmware and demonstrate power management algorithms.

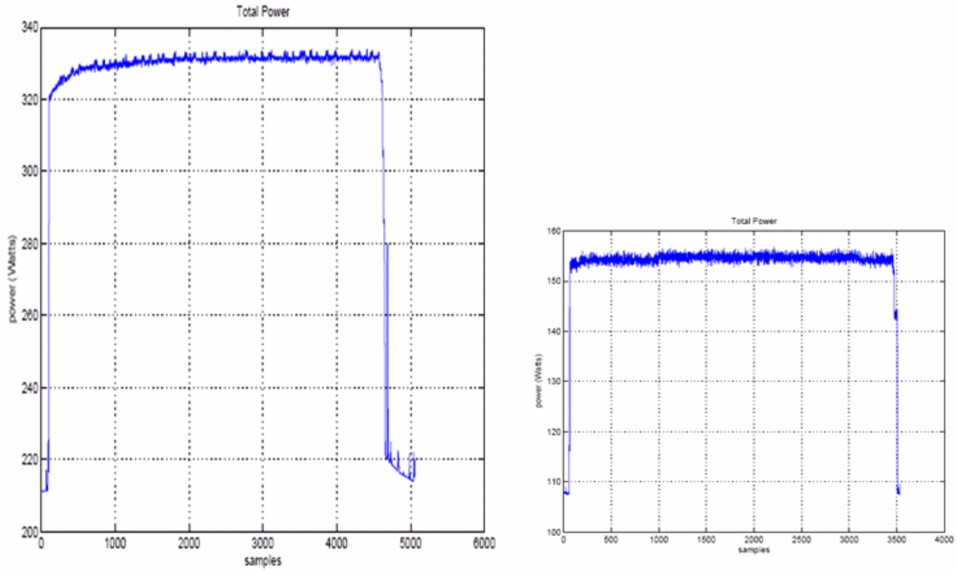


Figure 1: Total power measured on JS22 and HS21-XM for 437.leslie3d

Amester is executed from a remote machine and passes control commands through the BladeCenter management module to each blade it monitors. The service processor executes the commands by interfacing to the TPMD and returns the requested data. Amester is written entirely in TCL for fast prototyping of power management algorithms and for portability. It runs on any Windows or Linux system. The Tk graphics allow for easy visualization of data using strip charts in the BLT library. Amester can sample the power dissipation at intervals from 1ms on up, but 32ms to 256ms gives a good tradeoff between resolution and data volume collected. In the following experiments a sampling interval of 256ms to collect power measurements of blade total power, core power, and DIMM power as described above.

5.2 AEM

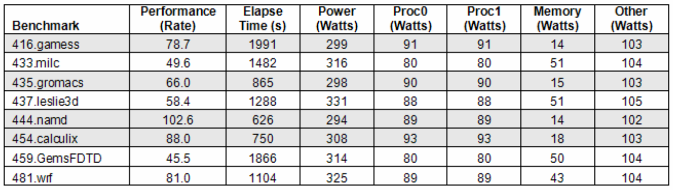
The Active Energy Manager (AEM) tool version 3.1, built on the Power Executive tool, provides management and control of the chassis and individual blade energy use [1]. It supports analysis and control such as power trending and capping, thermal trending, and CPU trending at the chassis or individual blade levels. The tool supports HS21-XM and JS22 blades.

6. Power and Performance Data and Models

The Amester tool was used to measure overall power consumption, core power consumption and DIMM power consumption as explained in Section 3. Figure 1 shows a typical total power consumption graph for JS22 and HS21-XM. Note that idle power sits at about 210 Watts for the JS22 and 107 Watts for HS21-XM, which is about two-thirds of the power consumed when the benchmark 437.leslie3D is running.

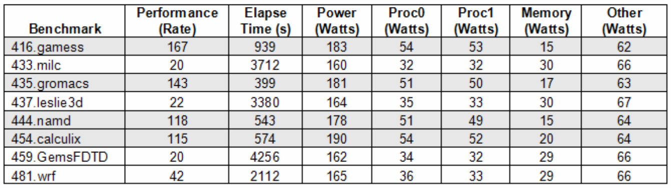
Tables 4, 5, and 6 below summarize the different components of power consumption per benchmark by platform. The different components are the processor sockets (labeled Proc 0 and Proc 1 for JS22 and HS21-XM), the memory DIMMS (labeled Memory for JS22 and HS21-XM) and Others for JS22 and HS21-XM which include the IO chip, off-chip memory controller and off-chip cache if they exist.

Table 4: Power and Performance on JS22 @ 4GHz



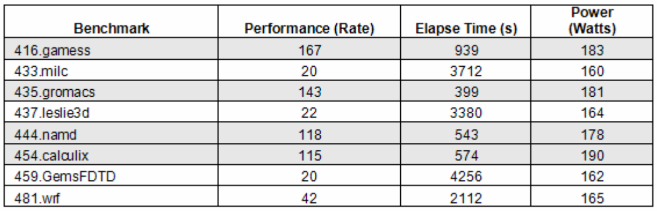
Benchmark	Performance (Rate)	Elapse Time (s)	Power (Watts)	Proc0 (Watts)	Proc1 (Watts)	Memory (Watts)	Other (Watts)
416.gamess	78.7	1991	299	91	91	14	103
433.milc	49.6	1482	316	80	80	51	104
435.gromacs	66.0	865	298	90	90	15	103
437.leslie3d	58.4	1288	331	88	88	51	105
444.namd	102.6	626	294	89	89	14	102
454.calculix	88.0	750	308	93	93	18	103
459.GemsFDTD	45.5	1866	314	80	80	50	104
481.wrf	81.0	1104	325	89	89	43	104

Table 5: Power and Performance on HS21-XM @ 2.8 GHz



Benchmark	Performance (Rate)	Elapse Time (s)	Power (Watts)	Proc0 (Watts)	Proc1 (Watts)	Memory (Watts)	Other (Watts)
416.gamess	167	939	183	54	53	15	62
433.milc	20	3712	160	32	32	30	66
435.gromacs	143	399	181	51	50	17	63
437.leslie3d	22	3380	164	35	33	30	67
444.namd	118	543	178	51	49	15	64
454.calculix	115	574	190	54	52	20	64
459.GemsFDTD	20	4256	162	34	32	29	66
481.wrf	42	2112	165	36	33	29	66

Table 6: Power and Performance on Nehalem @ 2.93 GHz



Benchmark	Performance (Rate)	Elapse Time (s)	Power (Watts)
416.gamess	167	939	183
433.milc	20	3712	160
435.gromacs	143	399	181
437.leslie3d	22	3380	164
444.namd	118	543	178
454.calculix	115	574	190
459.GemsFDTD	20	4256	162
481.wrf	42	2112	165

Due to the recent availability of Nehalem systems, we did not have sufficient time to port Amester on Nehalem. Thus we have only the total power consumption data for the Nehalem system gathered using a power meter (Table 6). As expected, processor power consumption accounts for the majority of the total power consumption. Coming in second, off-chip cache and IO chips consume a large amount of power regardless of the application execution characteristics.

We now derive a model to predict the power consumption of a given benchmark at frequency f_n given its characteristics measured at frequency f_0 and the platform characteristics measured at frequency f_n:

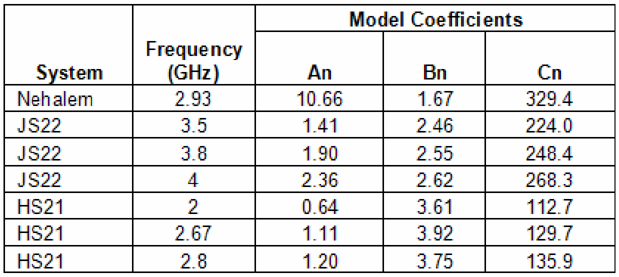
$$\mathbf{PWR(f_n) = A_n{*}GIPS(f_0) + B_n{*}GBS(f_0) + C_n} \qquad \mathbf{(1)}$$

where, PWR, GIPS and GBS are respectively power consumption, giga instructions per second, giga bytes per second at a given frequency. For workloads that are completely cache-contained, equation (1) may not be valid and may need additional terms to address cache activity.

GIPS(f_0) and GBS(f_0) are application characteristics measured at the nominal frequency (f_0). A_n, B_n and C_n are coefficients for a given platform at all possible clock frequencies, n, that the processor in the platform can be set. This model provides a better fit than using separate models for the other power, memory power, and core power, and then adding them up. The physical meaning is less evident in the combined model, but it is designed specifically to serve the purpose of projecting power at some frequency fn based on the nominal frequency f_0, thereby hiding the dependency of GIPS and GBS on clock frequency for a given benchmark. The model uses multiple linear regression analysis using the method of least squares for determining the power equation (1).

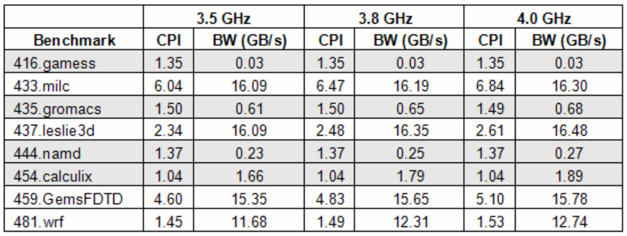
In Table 7, we present the resulting values of the A, B and C coefficients for the power equation. The average error using this model on all benchmarks over all machines is less than 1.6%. In Tables 8 and 9, we present CPI and total memory bandwidth measured for all workloads on JS22 and HS21-XM at different frequencies.

Table 7: Power equation coefficients calculated for all platforms



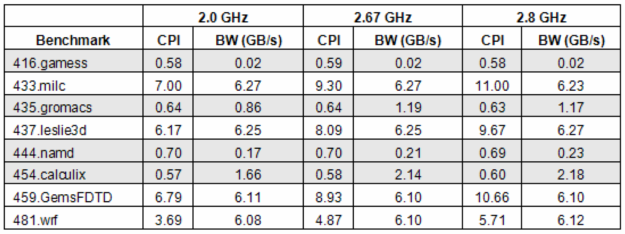
System	Frequency (GHz)	Model Coefficients		
		An	Bn	Cn
Nehalem	2.93	10.66	1.67	329.4
JS22	3.5	1.41	2.46	224.0
JS22	3.8	1.90	2.55	248.4
JS22	4	2.36	2.62	268.3
HS21	2	0.64	3.61	112.7
HS21	2.67	1.11	3.92	129.7
HS21	2.8	1.20	3.75	135.9

Table 8: CPI and Bandwidth measured on JS22



	3.5 GHz		3.8 GHz		4.0 GHz	
Benchmark	CPI	BW (GB/s)	CPI	BW (GB/s)	CPI	BW (GB/s)
416.gamess	1.35	0.03	1.35	0.03	1.35	0.03
433.milc	6.04	16.09	6.47	16.19	6.84	16.30
435.gromacs	1.50	0.61	1.50	0.65	1.49	0.68
437.leslie3d	2.34	16.09	2.48	16.35	2.61	16.48
444.namd	1.37	0.23	1.37	0.25	1.37	0.27
454.calculix	1.04	1.66	1.04	1.79	1.04	1.89
459.GemsFDTD	4.60	15.35	4.83	15.65	5.10	15.78
481.wrf	1.45	11.68	1.49	12.31	1.53	12.74

Table 9: CPI and Bandwidth measured on HS21-XM



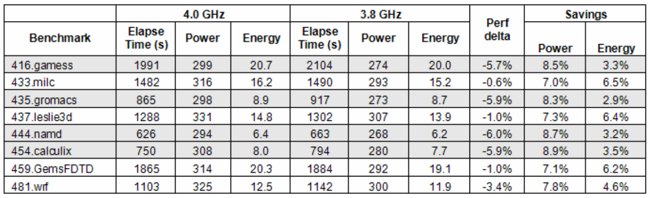
	2.0 GHz		2.67 GHz		2.8 GHz	
Benchmark	CPI	BW (GB/s)	CPI	BW (GB/s)	CPI	BW (GB/s)
416.gamess	0.58	0.02	0.59	0.02	0.58	0.02
433.milc	7.00	6.27	9.30	6.27	11.00	6.23
435.gromacs	0.64	0.86	0.64	1.19	0.63	1.17
437.leslie3d	6.17	6.25	8.09	6.25	9.67	6.27
444.namd	0.70	0.17	0.70	0.21	0.69	0.23
454.calculix	0.57	1.66	0.58	2.14	0.60	2.18
459.GemsFDTD	6.79	6.11	8.93	6.10	10.66	6.10
481.wrf	3.69	6.08	4.87	6.10	5.71	6.12

7. Impact of Frequency on Power and Energy

We now present the impact of frequency scaling on power and energy for the various benchmarks and platforms. The power and energy response of the benchmarks shown here provides an opportunity for a new scheduling mechanism and policy that can significantly reduce power and energy in high-performance computing systems.

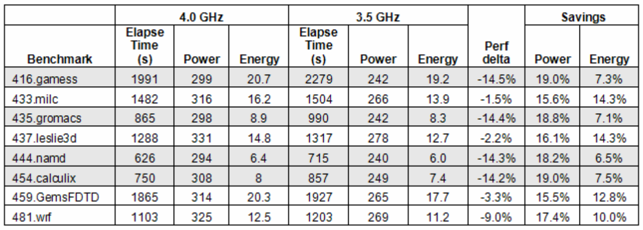
Tables 10 and 11 show the power and energy effects of down clocking, or reducing frequency, on the JS22. Table 12 shows the effects of over clocking, or increasing frequency, on the JS22. Similarly, Tables 13 through 15 show the effects of down clocking and over clocking on the HS21-XM.

Table 10: Down clocking from 4.0 to 3.8 GHz on JS22



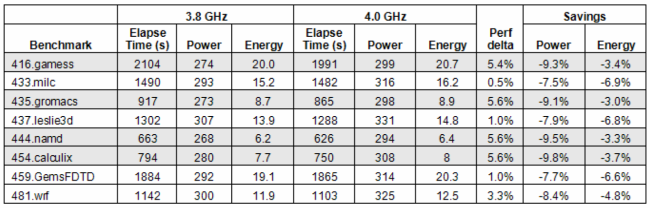
	4.0 GHz			3.8 GHz			Perf delta	Savings	
Benchmark	Elapse Time (s)	Power	Energy	Elapse Time (s)	Power	Energy		Power	Energy
416.gamess	1991	299	20.7	2104	274	20.0	-5.7%	8.5%	3.3%
433.milc	1482	316	16.2	1490	293	15.2	-0.6%	7.0%	6.5%
435.gromacs	865	298	8.9	917	273	8.7	-5.9%	8.3%	2.9%
437.leslie3d	1288	331	14.8	1302	307	13.9	-1.0%	7.3%	6.4%
444.namd	626	294	6.4	663	268	6.2	-6.0%	8.7%	3.2%
454.calculix	750	308	8.0	794	280	7.7	-5.9%	8.9%	3.5%
459.GemsFDTD	1865	314	20.3	1884	292	19.1	-1.0%	7.1%	6.2%
481.wrf	1103	325	12.5	1142	300	11.9	-3.4%	7.8%	4.6%

Table 11: Down clocking from 4.0 to 3.5 GHz on JS22



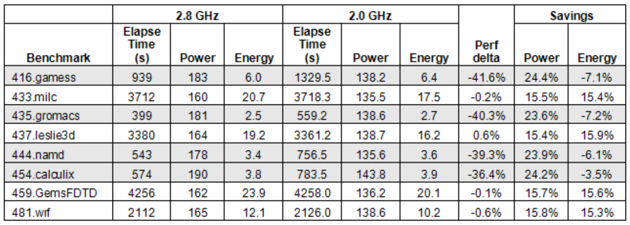
	4.0 GHz			3.5 GHz			Perf delta	Savings	
Benchmark	Elapse Time (s)	Power	Energy	Elapse Time (s)	Power	Energy		Power	Energy
416.gamess	1991	299	20.7	2279	242	19.2	-14.5%	19.0%	7.3%
433.milc	1482	316	16.2	1504	266	13.9	-1.5%	15.6%	14.3%
435.gromacs	865	298	8.9	990	242	8.3	-14.4%	18.8%	7.1%
437.leslie3d	1288	331	14.8	1317	278	12.7	-2.2%	16.1%	14.3%
444.namd	626	294	6.4	715	240	6.0	-14.3%	18.2%	6.5%
454.calculix	750	308	8	857	249	7.4	-14.2%	19.0%	7.5%
459.GemsFDTD	1865	314	20.3	1927	265	17.7	-3.3%	15.5%	12.8%
481.wrf	1103	325	12.5	1203	269	11.2	-9.0%	17.4%	10.0%

Table 12: Over clocking from 3.8 to 4.0 GHz on JS22



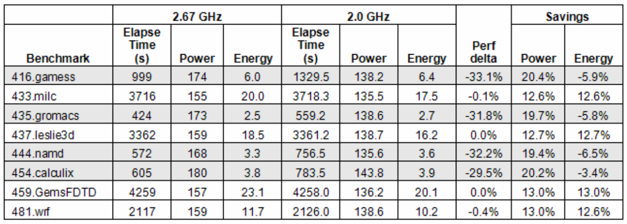
	3.8 GHz			4.0 GHz			Perf delta	Savings	
Benchmark	Elapse Time (s)	Power	Energy	Elapse Time (s)	Power	Energy		Power	Energy
416.gamess	2104	274	20.0	1991	299	20.7	5.4%	-9.3%	-3.4%
433.milc	1490	293	15.2	1482	316	16.2	0.5%	-7.5%	-6.9%
435.gromacs	917	273	8.7	865	298	8.9	5.6%	-9.1%	-3.0%
437.leslie3d	1302	307	13.9	1288	331	14.8	1.0%	-7.9%	-6.8%
444.namd	663	268	6.2	626	294	6.4	5.6%	-9.5%	-3.3%
454.calculix	794	280	7.7	750	308	8	5.6%	-9.8%	-3.7%
459.GemsFDTD	1884	292	19.1	1865	314	20.3	1.0%	-7.7%	-6.6%
481.wrf	1142	300	11.9	1103	325	12.5	3.3%	-8.4%	-4.8%

Table 13: Down clocking from 2.8 to 2.0 GHz on HS21-XM



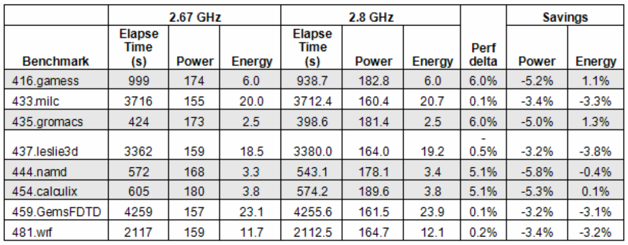
	2.8 GHz			2.0 GHz			Perf delta	Savings	
Benchmark	Elapse Time (s)	Power	Energy	Elapse Time (s)	Power	Energy		Power	Energy
416.gamess	939	183	6.0	1329.5	138.2	6.4	-41.6%	24.4%	-7.1%
433.milc	3712	160	20.7	3718.3	135.5	17.5	-0.2%	15.5%	15.4%
435.gromacs	399	181	2.5	559.2	138.6	2.7	-40.3%	23.6%	-7.2%
437.leslie3d	3380	164	19.2	3361.2	138.7	16.2	0.6%	15.4%	15.9%
444.namd	543	178	3.4	756.5	135.6	3.6	-39.3%	23.9%	-6.1%
454.calculix	574	190	3.8	783.5	143.8	3.9	-36.4%	24.2%	-3.5%
459.GemsFDTD	4256	162	23.9	4258.0	136.2	20.1	-0.1%	15.7%	15.6%
481.wrf	2112	165	12.1	2126.0	138.6	10.2	-0.6%	15.8%	15.3%

Table 14: Down clocking from 2.8 to 2.67 GHz on HS21-XM



	2.67 GHz			2.0 GHz			Perf delta	Savings	
Benchmark	Elapse Time (s)	Power	Energy	Elapse Time (s)	Power	Energy		Power	Energy
416.gamess	999	174	6.0	1329.5	138.2	6.4	-33.1%	20.4%	-5.9%
433.milc	3716	155	20.0	3718.3	135.5	17.5	-0.1%	12.6%	12.6%
435.gromacs	424	173	2.5	559.2	138.6	2.7	-31.8%	19.7%	-5.8%
437.leslie3d	3362	159	18.5	3361.2	138.7	16.2	0.0%	12.7%	12.7%
444.namd	572	168	3.3	756.5	135.6	3.6	-32.2%	19.4%	-6.5%
454.calculix	605	180	3.8	783.5	143.8	3.9	-29.5%	20.2%	-3.4%
459.GemsFDTD	4259	157	23.1	4258.0	136.2	20.1	0.0%	13.0%	13.0%
481.wrf	2117	159	11.7	2126.0	138.6	10.2	-0.4%	13.0%	12.6%

Table 15: Over clocking from 2.66 to 2.8 GHz on HS21-XM



	2.67 GHz			2.8 GHz			Perf delta	Savings	
Benchmark	Elapse Time (s)	Power	Energy	Elapse Time (s)	Power	Energy		Power	Energy
416.gamess	999	174	6.0	938.7	182.8	6.0	6.0%	-5.2%	1.1%
433.milc	3716	155	20.0	3712.4	160.4	20.7	0.1%	-3.4%	-3.3%
435.gromacs	424	173	2.5	398.6	181.4	2.5	6.0%	-5.0%	1.3%
437.leslie3d	3362	159	18.5	3380.0	164.0	19.2	-0.5%	-3.2%	-3.8%
444.namd	572	168	3.3	543.1	178.1	3.4	5.1%	-5.8%	-0.4%
454.calculix	605	180	3.8	574.2	189.6	3.8	5.1%	-5.3%	0.1%
459.GemsFDTD	4259	157	23.1	4255.6	161.5	23.9	0.1%	-3.2%	-3.1%
481.wrf	2117	159	11.7	2112.5	164.7	12.1	0.2%	-3.4%	-3.2%

In all the following tables, energy is defined as

Energy = Power*Elapse Time

From the tables, it can be seen that down clocking frequency on some platforms like the JS22 always saves power and energy regardless of the benchmark, while on other machines like the HS21-XM, down clocking always saves power but increases energy on the benchmarks with low memory bandwidth, as, for example, on 416.gamess, 435.gromacs, 444.namd, and 454.calculix.

This behavior arises when power saving is less than the performance degradation. This may happen for low memory bandwidth applications as the performance of these benchmarks is directly affected by CPU frequency. On the other hand, power saving on HS21-XM is less than power saving on JS22 since HS21 has a lower frequency processor. In other words down clocking has a bigger payback on high frequency platforms since the core power consumption is much higher (see Tables 4 and 5). Therefore, for platforms like the HS21-XM, over clocking can be an option for optimizing energy on low memory bandwidth applications**.**

8. Power and Energy-aware scheduling

Based on these observations, we propose a power and energy-aware job scheduling method. With traditional job schedulers, all jobs on a cluster are run at the same frequency as in Figure 2. The power and energy-aware scheduler, on the other hand, schedules parallel jobs such that all the nodes executing the various tasks or threads of one job are running at the same frequency, but different jobs may run at different frequencies, as shown in Figure 3.

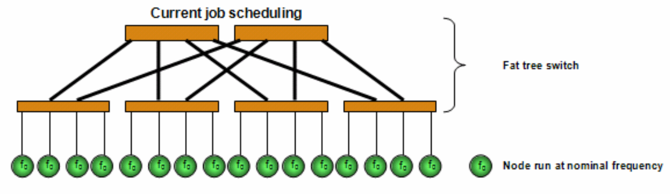


Figure 2: A cluster managed by a traditional scheduler

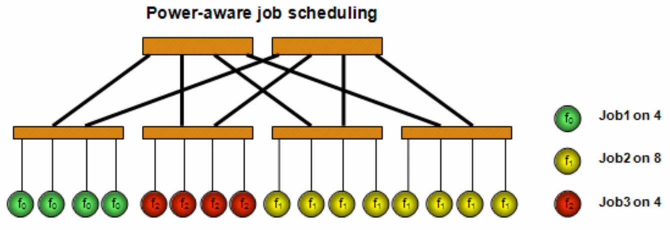


Figure 3: A cluster managed by the Power and Energy Aware scheduler

The total power consumption for the cluster, P, is used by the new scheduler to manage the maximum system power. P is provided by the administrator or by any external tool like AEM [4], and it can be static or dynamic. If P is not provided, it can be determined by the algorithm.

All nodes in the cluster have a power limit, p(i), i={1,..,n}, which can be different for each node and can be also static or dynamic. An admissible frequency is a frequency below this power limit for each node. Although each set of nodes will be running at a given frequency, power is managed globally at the cluster level.

The proposed algorithm has two phases. The first phase captures all the characteristics of the cluster in order to build the power and the performance models. This is done once at scheduler installation, when new components are introduced in the cluster, or when a new job is first executed on the system. Given the specific node and job characteristics, the first phase will also generate a report, used by the system administrator or a user that suggests the optimal approach to manage energy for a particular job on particular nodes.

The second phase determines the target frequency for a set of nodes when a user re-submits a job with the same characteristics and the job scheduling policy selected by the user/administrator.

In a first approach, we propose three scheduling policies: *Maximum performance*, *Minimum power within a maximum performance degradation and a maximum energy reduction*, and *Minimum energy within a maximum performance degradation*.All job policies set a frequency which is within the maximum power limits for each node. We now give example scheduling policies and the power and energy implications of these policies using the machines and data shown previously.

8.1 Maximum Performance Policy

On a JS22 running at 3.8 GHz, a low memory bandwidth application would see up to 5.6% performance gain with a power degradation of up to 9.8 % and energy degradation of up to 3.4 % if the clock frequency was raised to 4 GHz. A high memory bandwidth application would see no gain.

On an HS21-XM running at 2.67 GHz, a low bandwidth application would see up to 6% performance gain with a power degradation of up to 5 % and no energy reduction, or even a 1% energy saving, if the clock frequency was raised to 2.8 GHz. A high bandwidth application would see no gain.

8.2 Minimum Power Policy

On a JS22 running at 4GHz, a low bandwidth application would benefit with about 19% power saving, but with 15% performance degradation, if frequency was down clocked from 4 GHz to 3.5 GHz. A high bandwidth application would benefit with about 15% power saving but with a 3% performance degradation if frequency was down clocked from 4 GHz to 3.5 GHz.

On an HS21-XM running at 2.8 GHz, a low bandwidth application would benefit with about 25% power saving, but with 40% performance degradation, if frequency was down clocked from 2.8 GHz to 2.0 GHz. A high bandwidth application would benefit with about 15% power saving, but with 3% performance degradation, if frequency was down clocked from 2.8 GHz to 2.0 GHz.

8.3 Minimum Energy Policy

On a JS22 running at 4GHz, a low bandwidth application would benefit with about 7% energy saving, but with 15% performance degradation, if frequency was down clocked from 4.0 GHz to 3.5 GHz, or with about 4% energy saving, but with 6% performance degradation if frequency was down clocked from 4.0 GHz to 3.8 GHz A high bandwidth application would benefit with about 14% energy saving, but with up to 2% performance degradation, if frequency was down clocked from 4.0 GHz to 3.5GHz, or with 6% energy saving, but with up to 1% performance degradation, if frequency was down clocked from 4.0 GHz to 3.8 GHz.

On a HS21-XM running at 2.8 GHz, a low bandwidth application would lose up to about 7% energy, but with up to 40% performance degradation, if frequency was down clocked from 2.8 GHz to 2.0 GHz. A high bandwidth application would benefit with about 15% energy saving, but with 1% performance degradation, if frequency was down clocked from 2.8 GHz to 2.0 GHz.

As can be seen, some of those policies do not make sense from an energy standpoint, since, for example on the HS21-XM with low bandwidth applications, down clocking frequency would degrade performance much more than the associated power saving, leading to an increase of energy. But the policy may still be useful if a maximum power limit P must be maintained.

Based on these measurements and analysis, we therefore propose a single job scheduling policy to optimize power and energy. Based on the characteristics of both the application and the platform, the algorithm will either over clock or down clock the frequency in order to minimally degrade performance while still being within the total power limit P.

It will also provide the impact of this power and energy aware policy to the user/administrator in a report such that the percent of performance degradation and energy saving can be assessed and traded off as desired.

A couple of examples illustrate the reporting capabilities:

On JS22 running at 4GHz, a low bandwidth application would benefit by about 7% energy saving with about 15% performance degradation if the frequency was down clocked from 4.0 GHz to 3.5 GHz or about 4% energy saving with 6% performance degradation if the frequency was down clocked from 4.0 GHz to 3.8 GHz. A high bandwidth application would benefit by about 14% energy saving with up to 2% performance degradation if the frequency was down clocked from 4.0 GHz to 3.5GHz or by about 6% energy saving with up to 1%

performance degradation if the frequency was down clocked from 4.0 GHz to 3.8 GHz.

On an HS21-XM running at 2.67 GHz, a low bandwidth application would see no energy reduction or even a 1% energy saving with up to 6% performance gain and a power reduction of up to 5 % if the clock frequency was raised to 2.8 GHz. A high bandwidth application would benefit by about 12% energy saving with no performance degradation and a power saving of up to 13% if the frequency was down clocked from 2.67 GHz to 2.0 GHz.

9. Conclusions

This paper proposes a new job scheduler implementation that can optimize power and energy consumed by clusters when running parallel applications. The scheduler uses a simple multiple regression model to project the power consumed by a particular job, configure the characteristics of the nodes in the systems, and thereby maximize performance, minimize power, or minimize energy with a predictable impact on power, energy or performance.

The scheduler implements a model that predicts the power consumption and performance of a parallel HPC benchmark at any frequency based on performance metrics gathered when running the application the first time at a nominal frequency. We show that the model achieves an error less than 2% versus actual power-performance results.

Experimental data measured on different systems, including IBM Power6 and Intel Harpertown and Nehalem microprocessor based systems, are presented to validate the model, and assess the impact of the various scheduling policies. An example set of scheduling policies is presented based on the experiment results and an example administrator report is shown.

10. References

[1] H. Q. Le, W. J. Starke, J. S. Fields, F. P. O'Connell, D. Q. Nguyen, B. J. Rochetti, W. M. Sauer, E. M. Schwarz, M. T. Vaden, " IBM POWER6 microarchitecture, " *J. Res. & Dev.*, Vol. 51, No. 6, November 2007, pp. 639-662.

[2] M. S. Floyd, S. Ghiasi, T. W. Keller, K. Rajamani, F. L. Rawson, J. C. Rubio, and M. S. Ware, "System power management support in the IBM POWER6 microprocessor," *J. Res. & Dev.*, Vol. 51, No. 6, November 2007, pp. 733-746.

[3] H.-Y. McCreary, M. A. Broyles, M. S. Floyd, A. J. Geissler, S. P. Hartman, F. L. Rawson, T. J. Rosedahl, J. C. Rubio, and M. S. Ware, "EnergyScale for IBM POWER6 microprocessor-based systems," *J. Res. & Dev.*, Vol. 51, No. 6, November 2007, pp. 775-786.

[4] IBM Corporation, IBM BladeCenter, http://www.ibm.com/servers/eserver/bladecenter

[5] J. Allarey, V. George, S. Jihagirdar, "Power Management Enhancements in the 45nm Intel Core Microarchitecture," *Intel Technology Journal*, Vol. 12 Issue 3, 2008, pp. 169-178.

[6] K. Rajamani, H. Hanson, J. C. Rubio, S. Ghiasi, F. L. Rawson, "Online Power and Performance Estimation for Dynamic Power Management," IBM Research Technical Report, RC 24007, July 14, 2006.

[7] S. J. Lee, H. K. Lee, and P. C. Yew, "Runtime Performance Projection Model for Dynamic Power Management," *ACSAC 2007*, 2007, pp. 186-197.