THE INTERNATIONAL JOURNAL of HIGH PERFORMANCE COMPUTING APPLICATIONS

The International Journal of High Performance Computing Applications
25(3) 342–350

DOI: 10.1177/1094342011414749
hpc.sagepub.com
SAGE

Energy and performance characteristics of different parallel implementations of scientific applications on multicore systems

Charles Lively[1], Xingfu Wu[1], Valerie Taylor[1], Shirley Moore[2], Hung-Ching Chang[3] and Kirk Cameron[3]

Abstract

Energy consumption is a major concern with high-performance multicore systems. In this paper, we explore the energy consumption and performance (execution time) characteristics of different parallel implementations of scientific applications. In particular, the experiments focus on message-passing interface (MPI)-only versus hybrid MPI/OpenMP implementations for hybrid the NAS (NASA Advanced Supercomputing) BT (Block Tridiagonal) benchmark (strong scaling), a Lattice Boltzmann application (strong scaling), and a Gyrokinetic Toroidal Code – GTC (weak scaling), as well as central processing unit (CPU) frequency scaling. Experiments were conducted on a system instrumented to obtain power information; this system consists of eight nodes with four cores per node. The results indicate, with respect to the MPI-only versus the hybrid implementation, that the best implementation is dependent upon the application executed on 16 or fewer cores. For the case of 32 cores, the results were consistent in that hybrid implementation resulted in less execution time and energy. With CPU frequency scaling, the best case for energy saving was not the best case for execution time.

Keywords

energy consumption, frequency scaling, hybrid MPI/OpenMP, MPI, multicore system, performance characteristics, scientific applications

1 Introduction

Currently, the trend in high-performance computing (HPC) systems has shifted towards cluster systems with multicores. Energy consumption becomes a major challenge when using multicores to build petaflop or exaflop HPC systems (Kogge, 2008). Saving energy implies reducing power consumption or improving the performance (execution time), or both. The relationship between the performance and power consumption is non-linear and complex. In this paper, we investigate energy and performance characteristics of different parallel implementations of scientific applications on a multicore cluster system, and explore interactions between power consumption and performance.

The experiments conducted for this work utilized a multicore cluster, called Dori, from Virginia Tech with eight nodes, two AMD dual-core Opteron processors per node, and a power profiling tool, called PowerPack (Ge et al., 2010). We used PowerPack to measure the power consumption for our applications and Prophesy (Taylor et al., 2003) to measure the execution time of the applications. The experiments focus on exploring energy and performance characteristics for MPI-only versus hybrid MPI/OpenMP implementations (for three applications), and frequency scaling (for two applications). We use the following three applications for our experiments: a hybrid NASA Advanced Supercomputing (NAS) parallel benchmark BT (Block Tridiagonal) with Class B (based on NAS Parallel Benchmarks (NPB) BT 3.3) (Wu and Taylor, 2011); a Lattice Boltzmann application (Wu and Taylor, 2006); and a three-dimensional (3D) particle-in-cell application, Gyrokinetic Toroidal Code (GTC; Ethier, 2005; Wu and Taylor, 2009).

Our experimental results show that different methods can be utilized to improve performance and save energy.

[1]Department of Computer Science & Engineering, Texas A&M University, USA
[2]Department of Electrical Engineering and Computer Science, University of Tennessee-Knoxville, USA
[3]Department of Computer Science, Virginia Tech, USA

Corresponding author:
Xingfu Wu, Department of Computer Science & Engineering, Texas A&M University, MS 3112, College Station, TX 77843, USA
Email: wuxf@cs.tamu.edu

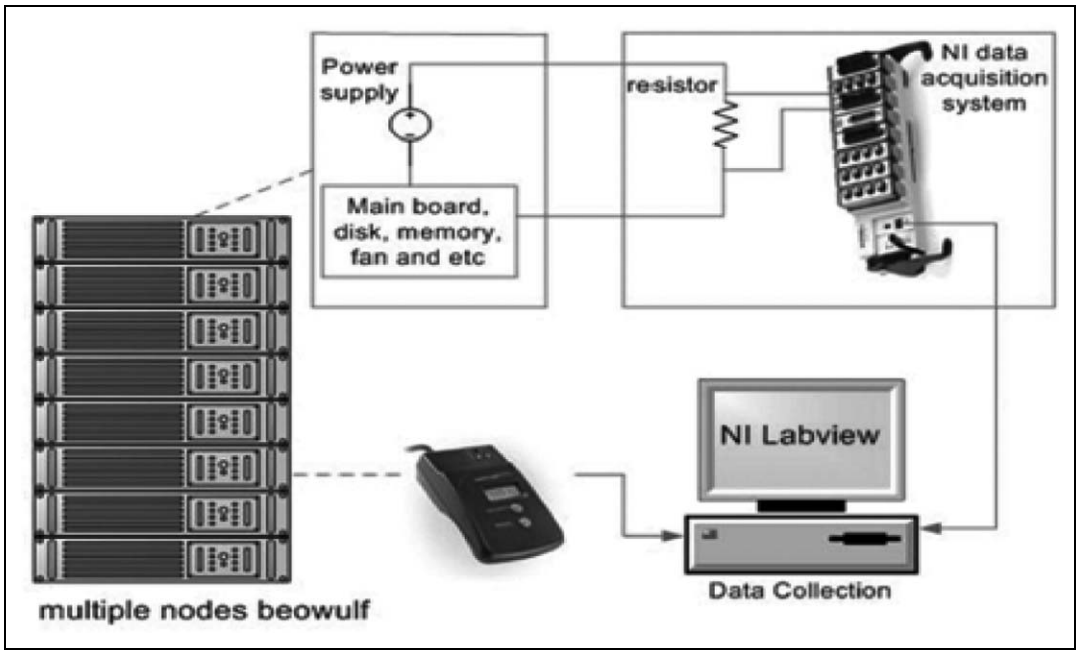

Figure 1. PowerPack framework (Ge et al., 2010).

The results indicate, with respect to the MPI-only versus the hybrid implementation, the best implementation is dependent upon the application for 16 or fewer cores. For the case of 32 cores, the results were consistent in that the hybrid implementation resulted in less execution time and energy. With frequency scaling, the best case for energy saving was not the best case for execution time.

The remainder of this paper is organized as follows. Section 2 describes the power profiling tool PowerPack and the experimental platform. Section 3 provides the experimental results with which we explore the energy and performance characteristics of different scientific applications. It is noted that for the remainder of the paper, we use the term performance to be synonymous with execution time. Section 4 discusses some related work. Section 5 summarizes the paper.

2 PowerPack and experiment platforms

Our experiments utilized an eight-node multicore system, Dori, which is available in the Department of Computer Science at Virginia Tech. Each node of the system consists of two dual-core AMD Opteron processors (1.8 GHz) and six 1 GB memory modules per node.

We used PowerPack (Ge et al., 2010), which provides power profiling information for advanced execution systems, to measure the power consumption for our applications running on the Dori cluster. The PowerPack framework shown in Figure 1 is a collection of software components, including libraries and application-programming interfaces (APIs), which enable system component-level power profiling correlated to application functions. PowerPack obtains measurements from power meters attached to the hardware of a system. The framework includes APIs and control daemons that use DVFS (dynamic voltage and frequency scaling) to enable energy reduction with very little impact on the performance of the system. As multicore systems evolve, the framework can be used to indicate the application parameters and the system components that affect the power consumption on the multicore unit. PowerPack allows the user to obtain direct measurements of the major system components' power consumption, including CPU, memory, hard disk, and motherboard. This fine-grain measurement allows power consumption to be measured on a per-component basis.

In this work, power consumption is measured on one main node and then remapped to other nodes on the system; this method is used because of the limited number of power measurement instruments. When executing an application, the PowerPack API and LabView data acquisition measurements are used to provide for fully automated application profiling of power consumption.

3 Experimental results

This section provides the details of our experimental results. In particular we explore MPI-only versus hybrid MPI/OpenMP implementations, and applying frequency scaling. In Section 3.1, we present the results for MPI-only versus hybrid MPI/OpenMP implementations for three applications. In Section 3.2, we present the results for applying frequency scaling to the NAS BT benchmark and the GTC application.

3.1. MPI and hybrid implementations

In this section, we use three scientific applications to explore the energy and performance of MPI-only versus

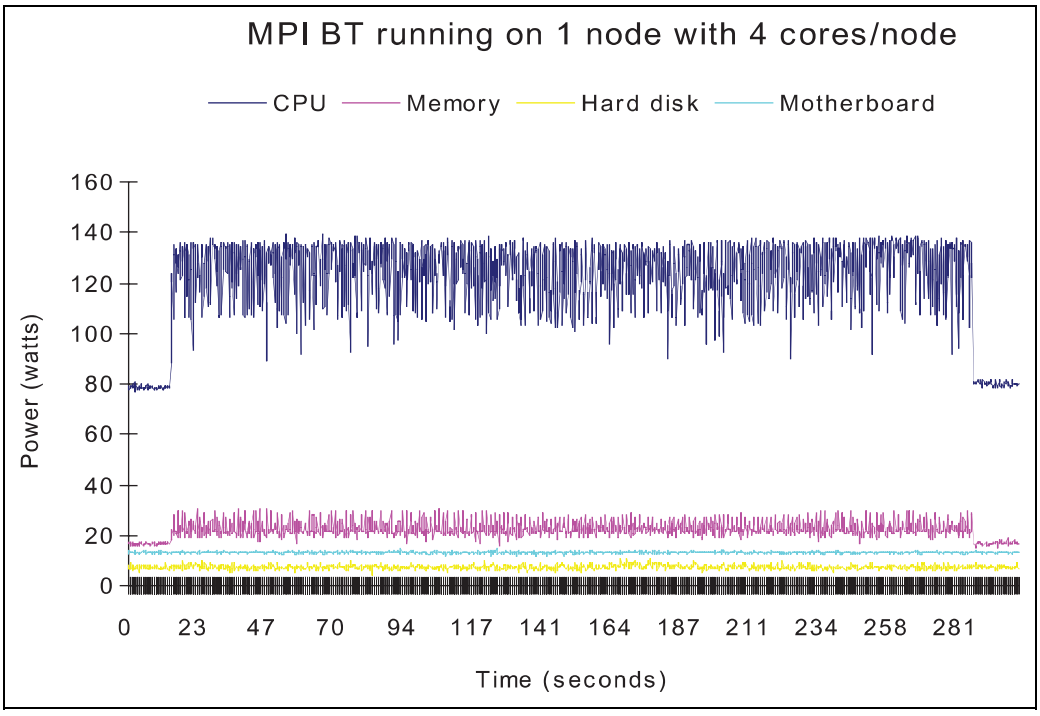

Figure 2. Power for the message-passing interface block tridiagonal (MPI BT) executed on a node with four MPI processes.

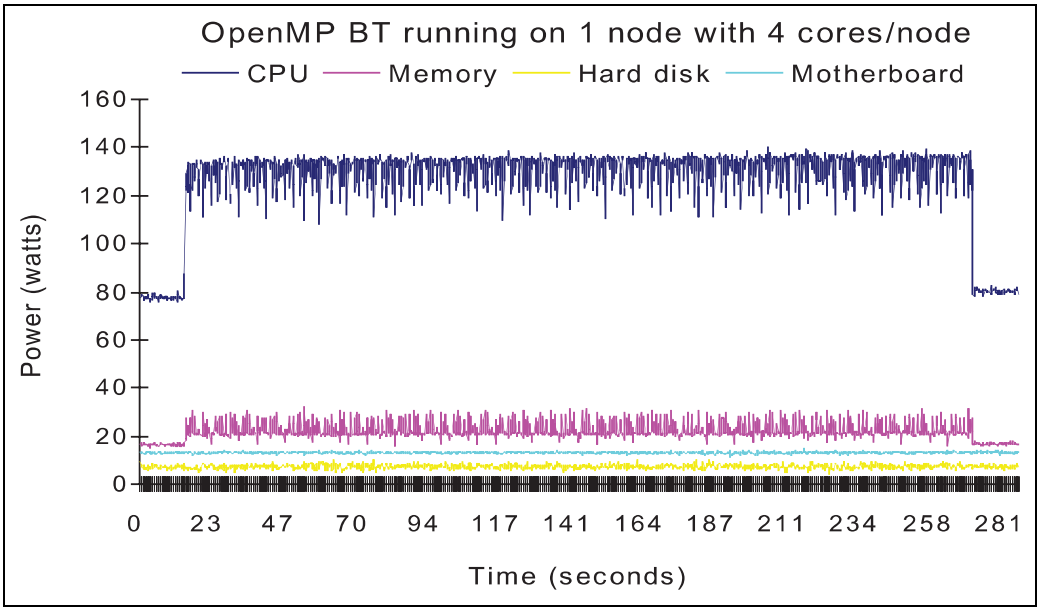

Figure 3. Power for message-passing interface (MPI)/OpenMP block tridiagonal (BT) executed on a node with one MPI process with four OpenMP threads.

hybrid (MPI/OpenMP) implementations: a hybrid NAS BT (Wu and Taylor, 2011), a Parallel Multiblock Lattice Boltzmann (PMLB) application and the GTC.

3.1.1. NAS parallel benchmark BT. We ran the hybrid MPI/OpenMP NPB BT with Class B on Dori to evaluate its performance and power consumptions for MPI and hybrid (MPI/OpenMP) implementations. Using the hybrid MPI/OpenMP programming can achieve better performance and also save energy. Our results show that the CPU and memory power consumption of the hybrid BT are higher than that for the MPI BT. Memory power consumption for the hybrid BT goes up and down significantly compared to the MPI BT, because of the use of shared address space by OpenMP.

We used PowerPack to collect power profiles for the CPU, memory, hard disk and motherboard for the MPI and hybrid MPI/OpenMP BT, as shown in Figures 2 and 3.

Table 1. Comparison for Block Tridiagonal (BT) for one and four nodes.

#Cores	BT type	Run time(s)	Total energy (J)
1×4	Hybrid	257 (–4.46%)	57,779 (–1.47%)
	MPI	269	58,643
4×4	Hybrid	71.723 (–5.84%)	15,941.091 (–4.56%)
	MPI	76.174	16,702.200

Figure 2 indicates that there are slacks when CPUs are waiting for data exchanges among all MPI processes. This causes CPU power to fluctuate up and down frequently for the MPI BT. Figure 3 illustrates that CPU power for the hybrid MPI/OpenMP BT does not vary as much as it does for the MPI BT, because the OpenMP threads take advantage of intra-node communication (shared address space), where we used one MPI process with four OpenMP threads for the execution of the hybrid BT. However, memory power consumptions for both are similar because of the relatively small problem size.

From Figures 2 and 3, we observe that the performance (execution time) for the hybrid MPI/OpenMP BT is slightly better than that for its MPI counterpart. The CPU power consumption for the hybrid BT is slightly higher than that for its MPI counterpart. The execution time for hybrid BT is 257 seconds and that for MPI BT is 269 seconds. The total energy consumption for the hybrid BT is 57,779 J; the energy consumption for MPI BT is 58,643 J. Table 1 provides a comparison of the MPI and hybrid performance of the application on four cores using one node in the system. Using the hybrid MPI/OpenMP implementation provides for an overall improvement in execution time of 4.46% and energy savings of 1.47%. Because the total energy consumption is the product of performance and power, 4.46% performance improvement and higher power consumption for the hybrid BT just results in the 1.47% energy saving.

Similarly, using the hybrid BT on four nodes not only can save 4.56% of the energy consumption, but also can achieve 5.84% performance improvement, as shown in Table 1. The 5.84% performance improvement and higher CPU and memory power consumption for the hybrid BT result in the 4.56% energy saving. This indicates that the hybrid parallel programming model MPI/OpenMP efficiently exploits the potential offered by the multicore cluster. The results are given for four and 16 cores only, because BT requires that the number of cores be a perfect square.

3.1.2. Parallel Multiblock Lattice Boltzmann. In this section, we discuss the energy performance of a large-scale scientific application, the PMLB (Wu and Taylor, 2006). The Lattice Boltzmann method is widely used in simulating fluid dynamics. It is based on kinetic theory, which entails a more fundamental level in studying the fluid than Navier–Stokes equations.

The PMLB application was implemented by researchers in the Aerospace Engineering Department at Texas A&M University using a MPI for communication. Our work provides for a hybrid implementation of the code incorporating OpenMP to take advantage of the shared-memory architecture of multicore chips.

The PMLB code demonstrates that the MPI-only implementation provides for a better performance in terms of execution time and energy consumption on up to 16 cores shown in Table 2. As the number of cores increases to 32 the execution time and energy consumption for the hybrid version becomes better than the MPI-only version. Specifically, on 32 cores (8×4), the energy consumption for the hybrid implementation is over 17% better than the MPI-only and the execution time for this parallel programming paradigm is 21% better.

The results are interesting in two ways. While energy is the product of power and execution time, the percentage reduction or increase for energy was not the same as that for performance. For example, with four cores, the execution time for the hybrid implementation was 33% larger, but the corresponding energy was 79% larger than MPI-only. Secondly, only when we have 32 cores is the hybrid method better. Further work is needed to explore if a different hybrid implementation would produce better results for 16 or fewer cores.

3.1.3. Gyrokinetic Toroidal Code. In this section, we discuss the energy performance of the GTC (Ethier, 2005; Wu et al., 2009). Note that the GTC is weak scaling with 100 particles per cell and 100 time steps.

Table 3 provides the energy and performance comparison of the GTC application executed on one to eight nodes of Dori with the default CPU frequency of 1.8 GHz, where KJ stands for thousand Joules, and $N \times M$ means N nodes with M cores per node. With the increase of the number of nodes from one to eight, the performance improvement percentage for the hybrid GTC over the MPI-only GTC increases from 37.22% on one node to 42.12% on eight nodes. In addition, the hybrid also saves 37.81% of the overall system energy over the MPI GTC on one node, and 41.86% of the total system energy on eight nodes. This also shows that using the hybrid MPI/OpenMP programming reduces the MPI communication overhead and achieves better performance and save energy.

It is interesting to observe that the performance improvement percentage and energy-saving percentage on a given number of nodes (from one to eight) are similar, mainly because the energy savings are the result of the performance improvement by the hybrid GTC. It indicates that power consumption for both the hybrid GTC and the MPI GTC is similar because the application is weak scaling. This is different from the results of the BT shown in Table 1, where the NAS BT is strong scaling and the performance improvement percentage for the hybrid BT is much larger

Table 2. Energy and performance comparison of message-passing interface (MPI) and hybrid Parallel Multiblock Lattice Boltzmann (PMLB) applications.

#Cores	PMLB type	Run time(s)	Total energy (KJ)	CPU energy (KJ)	Memory energy (KJ)	Disk energy (KJ)	Motherboard energy (KJ)
1×4	Hybrid	30.022 (33.92%)	6.337 (70.81%)	3.682 (65.55%)	0.818 (94.3%)	0.243 (27.89%)	0.411 (32.58%)
	MPI (baseline)	22.418	3.710	2.224	0.421	0.190	0.310
2×4	Hybrid	21.045 (18.74%)	8.629 (39.42%)	5.246 (40.61%)	0.916 (37.33%)	0.354 (19.59%)	0.584 (19.43%)
	MPI (baseline)	17.724	6.189	3.731	0.667	0.296	0.489
4×4	Hybrid	13.248 (5.78%)	10.534 (10.55%)	6.276 (12.17%)	1.229 (4.41%)	0.455 (10.44%)	0.738 (6.49%)
	MPI (baseline)	12.524	9.529	5.595	1.177	0.412	0.693
8×4	Hybrid	11.929 (–21.32%)	17.903 (–17.26%)	10.723 (–16.13%)	2.088 (–17.34%)	0.822 (–20.89%)	1.327 (–21.15%)
	MPI (baseline)	15.161	21.637	12.784	2.526	1.039	1.683

Table 3. Energy and performance comparison of message-passing interface (MPI) and hybrid Gyrokinetic Toroidal Code (GTC) applications.

#Cores	GTC type	Run time(s)	Total energy (KJ)	CPU energy (KJ)	Memory energy (KJ)	Disk energy (KJ)	Motherboard energy (KJ)
1×4	Hybrid	1302.773 (–37.22%)	270.223 (–37.81%)	162.969 (–38.52%)	27.086 (–33.47%)	9.699 (37.20%)	17.119 (–37.11%)
	MPI (baseline)	2075.376	434.524	265.071	40.714	15.445	27.221
2×4	Hybrid	1395.322 (–37.47%)	576.674 (–37.68%)	353.826 (–38.35%)	61.887 (–34.33%)	23.801 (–38.18%)	38.753 (–37.83%)
	MPI (baseline)	2231.652	925.401	574.003	94.238	38.501	62.333
4×4	Hybrid	1434.491 (–38.29%)	1182.959 (–38.40%)	711.065 (–39.31%)	118.186 (–34.64%)	41.824 (–39.26%)	74.670 (–38.58%)
	MPI (baseline)	2324.707	1920.578	1171.572	180.825	68.858	121.571
8×4	Hybrid	1463.457 (–42.12%)	2419.985 (–41.86%)	1457.945 (–42.39%)	244.013 (–37.73%)	86.806 (–41.7%)	153.596 (–41.80%)
	MPI (baseline)	2528.556	4162.998	2530.861	391.842	148.906	263.909

than its energy-saving percentage because of the higher power consumptions of the hybrid BT.

3.2. Performance and energy using frequency scaling

To perform frequency scaling on Dori, five frequency values are utilized. The default frequency and voltage for the system is set to 1.8 GHz and 1.4 V and can be adjusted to 1.0 GHz and 1.3 V. The CPU frequency on Dori can be adjusted in increments of 200 Hz from 1.8 to 1.0 GHz. We use the power profiling data of BT and GTC executed on four nodes (4×4) to further investigate the energy and performance impacts by using frequency scaling.

Table 4 provides the effects of applying frequency scaling to the NAS BT benchmark. When we scale down the CPU frequency from 1.8 to 1.0 GHz we observe that the hybrid BT has the minimum energy consumption of 14,444.036 J with the CPU frequency of 1.2 GHz. We use this energy consumption as a baseline to calculate the other percentages shown in Table 4. When increasing the CPU frequency from 1.2 to 1.8 GHz, we obtained performance improvement but lost energy. So there is a trade-off between performance and energy consumption. Achieving better performance may require using more energy.

Table 5 shows the energy and performance for the hybrid and MPI-only GTC at five CPU frequency gears from 1.8 to 1.0 GHz on the Dori system. This shows the effect that adjusting the frequency of the system has on the energy and performance of the application. As shown in Table 5, for the default CPU frequency of 1.8 GHz, the performance improvement percentage for the hybrid GTC over the MPI-only GTC is 38.29% on four nodes (with four cores per node), and the hybrid also saves 38.40% of the overall system energy over the MPI GTC on four nodes. We use the energy and performance for the MPI and hybrid GTC at the CPU frequency of 1.6 GHz as baseline to calculate the percentages of energy and performance at various frequencies, shown in Table 6. As we seek to explore the saving in energy we use the lowest energy consumption obtained at 1.6 GHz as the baseline.

For the given problem size and number of cores, it is obvious to see the total application execution times for both

Table 4. Comparison for message-passing interface (MPI) and hybrid Block Tridiagonal (BT) applications on 4×4 (16 cores) using frequency scaling.

CPU speed	BT type	Run time(s)	Total energy (KJ)	CPU energy (KJ)	Memory energy (KJ)	Disk energy (KJ)	Motherboard energy (KJ)
1.8 Ghz	Hybrid	71.723 (–25.31%)	15,941.091 (10.36%)	9453.668 (22.88%)	1580.718 (–26.76%)	508.679 (–30.83%)	919.018
	MPI	76.174 (–27.82%)	16,702.200 (15.63%)	9986.521	1765.706	554.347	997.488
1.6 Ghz	Hybrid (baseline)	76.139 (–21.80%)	15,058.230 (4.25%)	8737.304 (13.57%)	1713.132 (–20.62%)	566.728 (–22.94%)	1000.655
	MPI	81.841 (–15.94%)	15,903.052 (10.1%)	9088.220	1858.386	598.208	1062.258
1.4 Ghz	Hybrid	84.849 (–12.86%)	14,732.076 (1.99%)	8186.828 (6.41%)	1852.877 (–14.15%)	601.683 (–18.19%)	1091.145
	MPI	90.530 (–7.02%)	15,624.080 (8.17%)	8577.551	1992.754	639.778	1163.661
1.2 Ghz	Hybrid	97.366	14,444.036	7693.547	2158.369	735.495	1288.510
	MPI	101.990 (4.74%)	15,088.793 (4.46%)	8101.081	2330.107	803.493	1372.160
1.0 Ghz	Hybrid	111.947 (14.97%)	17,041.246 (17.98%)	9325.778 (21.22%)	2480.800 (14.93%)	873.530 (18.77%)	1503.207
	MPI	117.394 (20.56%)	17,774.750 (23.06%)	9630.939	2606.256	898.152	1559.354

Table 5. Gyrokinetic Toroidal Code (GTC) power profiling on 4×4 (16 cores) using frequency scaling.

CPU speed	GTC type	Run time(s)	Total energy (KJ)	CPU energy (KJ)	Memory energy (KJ)	Disk energy (KJ)	Motherboard energy (KJ)
1.8 Ghz	Hybrid	1434.491 (–8.62)	1182.959 (–3.72%)	711.065 (–7.1%)	118.186 (–8.09%)	41.824 (10.81%)	74.669 (–9.26%)
	MPI	2324.707 (48.1%)	1920.578 (68.50%)	1171.572	180.825	68.858	121.571
1.6 Ghz	Hybrid (baseline)	1569.960	*1139.831*	664.098	128.594	46.894	82.292
	MPI	2511.532 (59.97%)	2057.516 (80.51%)	1253.041	196.440	76.902	133.030
1.4 Ghz	Hybrid	1773.444 (12.96%)	1143.615 (0.03%)	661.161 (–0.04%)	153.450 (19.39)	59.649 (27.19%)	98.017 (19.10%)
	MPI	2791.607 (77.81%)	1778.682 (8.00%)	1040.457	230.353	93.187	153.778
1.2 Ghz	Hybrid	2094.598 (33.40%)	1162.393 (1.97%)	628.386 (–5.37%)	176.897 (37.56%)	68.966 (47.1%)	114.914 (39.64%)
	MPI	3126.446 (99.1%)	1724.057 (51.26%)	940.227	254.275	103.819	171.746
1.0 Ghz	Hybrid	2445.155 (37.87%)	1393.650 (22.26%)	769.366 (15.85%)	204.96 (4.34%)	81.417 (73.61%)	134.758 (63.76%)
	MPI	3553.982 (127.37%)	2015.483 (76.82%)	1112.277	285.326	115.870	193.778

Table 6. Function-level performance (seconds) comparison of Gyrokinetic Toroidal Code (GTC) using frequency scaling.

CPU speed	GTC type	Run time	Pusher	Shift	Charge	Poisson	Smooth	Field	Load
1.8 Ghz	Hybrid	1434.491	854.5	36.75	498.1	16.94	10.25	4.289	8.773
	MPI	2324.707	823.7	268.4	627.7	159.6	284.9	147.3	8.899
1.6 Ghz	Hybrid	1569.960 (–9.44%)	940.1 (–10.02%)	39.64 (–7.83%)	542.3 (–8.87%)	20.81 (–22.84%)	8.019 (27.82%)	4.389 (–2.33%)	9.693 (–10.49%)
	MPI	2511.532 (–7.44%)	850.7	348.2	692.5	160.8	292.7	151.7	11.55
1.4 Ghz	Hybrid	1773.444 (–23.62%)	1067.00 (–24.87%)	38.48 (–4.8%)	617.2 (–23.91%)	21.01 (–24.03%)	8.532 (20.13%)	4.680 (–9.11%)	10.82 (–23.33)
	MPI	2791.607 (–20.10%)	1053.00	307.40	772.6	157.8	323.7	163.2	10.57
1.2 Ghz	Hybrid	2094.598 (–46.02%)	1255.00 (–46.87%)	46.43 (–26.34%)	734.2 (–47.4%)	24.21 (–42.91%)	9.884 (3.7%)	5.517 (–28.63)%	12.36 (–40.88%)
	MPI	3126.446 (34.49%)	1218.00	324.3	897.9	162.80	335.00	171.2	11.94
1.0 Ghz	Hybrid	2445.155 (–70.45%)	1473.00 (–72.38%)	48.18 (–31.10%)	855.9 (–71.83%)	28.60 (–68.83%)	11.21 (–9.36%)	5.879 (–37.1%)	14.51 (–65.39%)
	MPI	3553.982 (–52.88%)	1458.00	339.0	1056.00	161.3	345.9	171.3	17.02

he MPI and hybrid GTC increase with decreasing the CPU frequency from 1.8 to 1.0 GHz, as shown in Table 5. For instance, the execution time for the hybrid GTC executed on four nodes increases up to 37.87% when decreasing the CPU frequency to 1.0 GHz. Decreasing CPU frequency means that a lower voltage is utilized. This results in lower power consumption for the application. However, because the energy is the product of power consumption and the execution time, from Table 5, we observe that the total energy consumption for the hybrid GTC decreases 3.78% for the frequency of 1.6 GHz, 3.40% for the frequency of 1.4 GHz, and 1.77% for the frequency of 1.2 GHz, but increases 17.81% for the frequency of 1.0G Hz; the total energy consumption for the MPI GTC increases 7.13% for the frequency of 1.6G Hz, decreases 8.00% for the frequency of 1.4 GHz, and decreases 11.4% for the frequency of 1.2 GHz, but increases 4.94% for the frequency of 1.0 GHz. So there is performance–energy trade-off that needs to be seriously considered when applying frequency scaling to an application.

Table 6 illustrates the effect that frequency scaling has on the performance of GTC at a functional granularity. The run times for the MPI and hybrid GTC at the default frequency of 1.8 GHz are used as baselines to calculate the performance percentages for reduced frequencies. We observe that the hybrid GTC outperforms its MPI counterpart because of large performance improvements for the five functions, *shift*, *charge*, *Poisson*, *smooth*, and *field* of the GTC and poor L2 cache behavior for the MPI implementation, which increases the amount of off-chip communications and degrades the performance. This is consistent across different CPU frequencies. This further shows that using the hybrid MPI/OpenMP programming can not only reduce MPI communication overhead, but also achieve better performance and save energy. The function-level information for CPU frequency scaling can help us in finding the best combination of CPU frequency adjustments for the entire GTC to save more energy when applying frequency scaling to the entire application.

4 Related work

HPC researchers have developed several techniques and systems with the goal of improving the power consumption and energy utilization of scientific applications. Power-aware infrastructures provided by Hsu and Feng (2005), Song et al. (2009), and Ge et al. (2010) are able to capture the energy consumption of scientific applications on large-scale parallel platforms. In this work, the PowerPack discussed in Ge et al. (2010) is utilized.

DVFS is a well-used energy reduction technique (Freeh et al., 2005, 2008). In Freeh et al. (2005), the authors used DVFS to reduce the energy consumption of MPI applications by determining different phases for the application. This work utilized a brute-force approach to determine the optimal energy-performance setting for each phase and then execute the application accordingly. The NAS BT benchmark is divided into two phases that are executed at multiple gear points. The BT is executed at gears 1 and 2, giving an energy saving of 10% with a time penalty of 5%. In Freeh et al. (2008), a system called Jitter was introduced with the goal of exploiting the MPI wait time for load-imbalanced applications. The frequency and voltage of nodes with less computational time were reduced to save overall energy while other compute-intensive nodes completed.

In Rountree et al. (2009), a run-time system, *Adagio*, was presented to combine static methods for energy reduction methods, such as dynamic voltage scaling (DVS), with scheduling algorithms to reduce energy without dramatically increasing the overall execution time. The run-time mechanism used was for slack prediction and was applied to three different application codes: UMT2k, ParaDis, and the NPB. Adagio slows computation that is off the critical path so it does not affect execution time significantly.

In Song et al. (2011), an iso-efficiency energy model is proposed to explore the application and machine characteristics that enable balanced energy use and performance. The iso-efficiency energy model was able to produce prediction errors less than 5% for various MPI applications from the NPB.

In Curtis-Maury et al. (2008), a prediction model is introduced to provide energy savings on multithreaded application programs. This work introduced the ACTOR system, which allows for dynamic control of active threads in an application to save energy. In this work the authors use dynamic concurrency throttling (DCT) to save power by rendering some cores on their multicore system idle. The work focused on the performance characteristics of the OpenMP versions from NPB (version 3.1) to achieve performance improvements of 17.9% and energy savings of 26.7%.

As most research work in this area focuses on MPI or multithreaded scientific applications, the current trend of multicore processors used in large-scale computing systems makes hybrid parallel programming models, such as hybrid MPI/OpenMP, more popular for large-scale scientific applications. In (Li et al., 2010), the authors use DVFS and DCT to reduce the energy requirements of hybrid application codes for several benchmarks. A power-aware performance predictor is used to develop a power-efficient algorithm for ASC (Advanced Simulation and Computing) equoia benchmarks and NPB multizone (MZ) benchmarks. They achieved energy savings from 4.1% up to 13.8% with negligible performance loss. Our work presented in this paper differs from the previous work in that we investigate the energy-performance benefits of hybrid MPI/OpenMP applications over MPI-only applications on multicore systems from parallel programming models' perspective, because a hybrid MPI/OpenMP program is not only able to achieve multiple levels of parallelism, but also is able to reduce the communication overhead of MPI within a multicore node, by taking advantage of the shared address space and on-chip high inter-core bandwidth and low inter-core latency provided by multicore clusters.

5 Conclusions

In this paper, we investigated energy and performance characteristics of different parallel implementations of scientific applications on multicore systems, and explored interactions between power consumption and application performance. We used the power profiling tool PowerPack to collect power profiling data for three scientific applications: a hybrid NAS parallel BT benchmark, a hybrid Lattice Boltzmann application PMLB, and a hybrid GTC, for our comparative analysis of energy and performance on multicore clusters. Our experimental results show that there are various ways to save energy and improve performance of parallel application code. Firstly, we found, with respect to the MPI-only versus the hybrid implementation for a scientific application, the best implementation is dependent upon the application executed on 16 or fewer cores. For the case of 32 cores, the results were consistent in that the hybrid resulted in less execution time and energy. For example, the hybrid PMLB achieved 21% performance improvement and 17% reduction in energy consumption compared to the MPI-only implementation. With the CPU frequency scaling, the best case for energy saving was not the best case for execution time. For example, the hybrid GTC executed at the CPU frequency 1.6 GHz provided the lowest energy consumption but the execution time increased by 8.62%.

Our hybrid implementations are based on the existing MPI applications and were implemented to exploit the shared-memory architectures of multicore systems. For further work, we are working on developing analytical models for energy and performance at different levels based on our experimental results in this paper, and will investigate the energy and performance of these applications and additional scientific applications on other multicore systems as PowerPack becomes available on different systems.

This work is part of the National Science Foundation (NSF)-funded MuMI (Multicore application Modeling Infrastructure) Project (http://www.mumi-tool.org), which facilitates the systematic measurement, modeling, and prediction of performance, power consumption, and performance–power trade-offs for multicore systems. In the future, we will use the MuMI to model, analyze, and optimize performance and power consumption of these benchmarks and applications on multicore systems as a starting point.

Acknowledgements

The authors would like to acknowledge Stephane Ethier from Princeton Plasma Physics Laboratory for providing the GTC code and Research Experience for Undergraduates (REU) summer student Ashraf Bah Rabiou for his work on the hybrid PMLB.

Funding

This work was supported by the NSF (grant numbers CNS-0911023, CNS-0910899, CNS-0910784, and CNS-0905187).

Conflict of interest statement

None declared.

References

Curtis-Maury M, Blagojevic F, Antonopoulos CD and Nikolopoulos DS (2008) Prediction-based power-performance adaptation of multithreaded scientific codes. *IEEE Trans Parallel Distrib Syst* 19(10): 1396-1410.

Ethier S (2005) First Experience on BlueGene/L, *BlueGene Applications Workshop*, ANL, 27–28 April. Available at: http://www.bgl.mcs.anl.gov/Papers/GTC_BGL_20050520.pdf

Freeh V, Kappiah N, Lowenthal D and Bletsch T (2008) Just-in-time dynamic voltage scaling: exploiting inter-node slack to save energy in MPI programs. *J Parallel Distrib Comput* 68: 1175-1185.

Freeh V, Pan F, Lowenthal D and Kappiah N (2005) Using multiple energy gears in MPI programs on a power-scalable cluster. In: *Proceedings of the 10th ACM Symposium on Principles and Practice of Parallel Programming (PPOPP)*, June.

Ge R, Feng X, Song S, Chang H, Li D and Cameron K (2010) PowerPack: energy profiling and analysis of high-performance systems and applications. *IEEE Trans Parallel Distrib Syst* 21: 658-671.

Hsu C-H and Feng W-C (2005) A power-aware run-time system for high-performance computing. In: *Proceedings of the IEEE/ACM Supercomputing 2005 (SC05)*, November.

Li D, de Supinski B, Schulz M, Cameron K and Nikolopoulos DS (2010) Hybrid MPI/OpenMP power-aware computing. In: *Proceedings of the 24th International Parallel and Distributed Processing Symposium (IPDPS)*, Atlanta, GA, April.

Kogge PM (ed.) (2008) Exascale computing study: technology challenges in achieving exascale systems. *CSE Dept. Tech. Report TR-2008-13*, University of Notre Dame, 28 September.

Rountree B, Lowenthal D, et al. (2009) Adagio: making DVS practical for complex HPC applications. In: *Proceedings of the 23rd International Conference on Supercomputing (ICS09)*, New York.

Song S, Ge R, Feng X and Cameron K (2009) Energy profiling and analysis of the HPC challenge benchmarks. *Int J High Perform Comput Appl* 23: 265-276.

Song S, Su C, Ge R, et al. (2011) Iso-energy-efficiency: an approach to power-constrained parallel computation. In: *Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*.

Taylor V, Wu X and Stevens R (2003) Prophesy: an infrastructure for performance analysis and modeling system of parallel and grid applications. *ACM SIGMETRICS Perform Eval Rev* 30: 13-18.

Wu X and Taylor V (2011) Performance characteristics of hybrid MPI/OpenMP implementations of NPB SP and BT on large-scale multicore supercomputers. *ACM SIGMETRICS Perform Eval Rev* 38: 56-62.

Wu X, Taylor V, Lively C and Sharkawi S (2009) Performance analysis and optimization of parallel scientific applications on CMP clusters. *Scalable Comput Pract Exper* 10: 61-74.

Wu X, Taylor V, Garrick S, Yu D and Richard J (2006) Performance analysis, modeling and prediction of a parallel multiblock lattice Boltzmann application using prophesy system. In: *Proceedings of the IEEE International Conference on Cluster Computing*, 25–28 September.

Author's Biographies

Charles Lively is a PhD candidate in the Department of Computer Science and Engineering working with Valerie E Taylor. He received his BSE in Computer Engineering from Mercer University and MS in Computer Engineering from Texas A&M University. His research interests include high-performance computing with special interest in the analysis and modeling of scientific applications.

Xingfu Wu has been working at Texas A&M University as TEES (Texas Engineering Experiment Station) Research Scientist since July 2003. He is a senior ACM (Association for Computing Machinery) member and an Institute of Electrical and Electronics Engineers (IEEE) member. He received his BS and MS in Mathematics from Beijing Normal University and his PhD in computer science from the Beijing University of Aeronautics and Astronautics. His research interests are performance evaluation and modeling, parallel and grid computing, and power and energy analysis in HPC systems. His monograph, *Performance Evaluation, Prediction and Visualization of Parallel Systems*, was published by Kluwer Academic Publishers (ISBN 0-7923-8462-8) in 1999.

Valerie Taylor earned her BS in Electrical and Computer Engineering and MS in Computer Engineering from Purdue University in 1985 and 1986, respectively, and her PhD in Electrical Engineering and Computer Science from the University of California, Berkeley, in 1991. From 1991 to 2002, she was a member of the faculty in the Electrical and Computer Engineering Department at Northwestern University. She joined the faculty of Texas A&M University as Head of the Dwight Look College of Engineering's Department of Computer Science in January of 2003, and is, also, currently a holder of the Royce E. Wisenbaker Professorship. Her research interests are in the area HPC. She has authored or co-authored over 100 papers in these areas. Dr Taylor is a member of ACM and Senior Member of IEEE-CS.

Shirley Moore received her PhD in Computer Sciences from Purdue University in 1990. She currently holds a position as a Research Associate Professor in the Electrical Engineering and Computer Science Department at the University of Tennessee. Her research interests are in performance modeling and analysis of scientific applications and in performance analysis tools. She is a member of the development team of the widely used Performance Application Programming Interface (PAPI) library for accessing hardware performance counters.

Hung-Ching Chang is a PhD candidate in the Department of Computer Science at Virginia Polytechnic Institute and State University. He received his BE in Electronic Engineering from Huafan University, Taiwan. His research interests include parallel distributed systems, energy-efficient computing, HPC, and performance modeling and analyses. He is a student member of the IEEE.

Kirk Cameron is an associate professor of Computer Science at Virginia Polytechnic Institute and State University. He holds a BS from the University of Florida and a PhD in Computer Science from Louisiana State University. He pioneered the area of high-performance power-aware computing and has received numerous research awards, including NSF and DOE (Department of Energy) Career Awards and the IBM Faculty Award. He is a research fellow of the Virginia Tech College of Engineering and the Uptime Institute. He also serves on the editorial board for *IEEE Computer*.