Performance and Energy Efficiency Analysis of 64-bit ARM Using GAMESS

Ananta Tiwari*† Kristopher Keipert‡ Adam Jundt† Joshua Peraza† Sarom S. Leang‡
Michael Laurenzano†§ Mark S. Gordon‡ Laura Carrington*†

*Performance Modeling and Characterization (PMaC) Laboratory,
San Diego Supercomputer Center, La Jolla, CA, USA

‡Iowa State University, Ames, IA, USA
kris@si.msg.chem.iastate.edu, {sok1, mgordon}@iastate.edu

§University of Michigan, Ann Arbor, MI, USA

† EP Analytics, Inc., San Diego, CA
{ananta.tiwari, adam.jundt, joshua.peraza, michaell, laura.carrington}@epanalytics.com

ABSTRACT

Power efficiency is one of the key challenges facing the HPC co-design community, sparking interest in the ARM processor architecture as a low-power high-efficiency alternative to the high-powered systems that dominate today. Recent advances in the ARM architecture, including the introduction of 64-bit support, have only fueled more interest in ARM. While ARM-based clusters have proven to be useful for data server applications, their viability for HPC applications requires an in-depth analysis of on-node and inter-node performance. To that end, as a co-design exercise, the viability of a commercially available 64-bit ARM cluster is investigated in terms of performance and energy efficiency with the widely used quantum chemistry package GAMESS. The performance and energy efficiency metrics are also compared to a conventional x86 Intel Ivy Bridge system. A 2:1 Moonshot core to Ivy Bridge core performance ratio is observed for the GAMESS calculation types considered. Doubling the number of cores to complete the execution faster on the 64-bit ARM cluster leads to better energy efficiency compared to the Ivy Bridge system; i.e., a 32-core execution of GAMESS calculation has approximately the same performance and better energy-to-solution than a 16-core execution of the same calculation on the Ivy Bridge system.

1. INTRODUCTION

The energy consumption of large scale high performance computing (HPC) systems is becoming an increasing concern as the computational science community heads towards the Exascale era. Current estimates indicate that processor efficiencies will have to evolve from the current 5 GFLOPS/Watt to 50 GFLOPS/Watt for Exascale machines to be viable. While multiple processor architectures are being considered in the pursuit of more energy-efficient HPC, it is almost certain that the ARM architecture will figure prominently into the solution set, as evidenced by its presence in the Mont-Blanc project [30]. Furthermore, the entire HPC market is only a small fraction of the total computing market and, therefore, relies primarily on commodity processor technology. In mobile computing, the ARM architecture plays a large role as the most ubiquitous energy-efficient processor architecture available. Consequently, investors and companies have recognized the potential of the ARM architecture in HPC and enterprise computing. For example, HP, Dell, and others are fielding enterprise-class servers based on ARM processors. NVIDIA has developed a hybrid ARM-GPGPU system on chip (SoC) which was initially deployed at the Barcelona Supercomputing Center for the Mont-Blanc project.

Given the rapid penetration of ARM systems into the HPC market, it is critical that the performance and energy efficiency of ARM is understood in the context of well-established architectures used in HPC. Towards that end, the performance, parallel efficiency, and power efficiency of the modern ARM-based HP Moonshot system is analyzed, and compared to an Intel Ivy Bridge system. As a co-design exercise, this study is conducted for the prominent large scale computational chemistry package GAMESS [32]. Of primary interest is the investigation of which types of computations within the GAMESS package see the best performance scaling and energy-efficiency on the Moonshot system, in addition to the identification of architectural features that bottleneck the performance. The Intel Ivy Bridge system is used as a reference architecture for the bottleneck analysis.

The paper is organized as follows: First, descriptions are provided for the GAMESS software package and for each

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
Co-HPC November 15-20, 2015, Austin, TX, USA
Copyright 2015 ACM 978-1-4503-3992-6/15/11
http://dx.doi.org/10.1145/2834899.2834905 ...$15.00.

type of GAMESS calculation explored in this study. In Section 3, the methodology used for system benchmarking is described. This description includes details about how the metrics relevant to the performance and energy efficiency of the target systems are measured. In Section 4, the results which demonstrate the scalability and energy efficiency of various types of chemistry calculations on the Moonshot system are presented. The Moonshot system is then compared to Ivy Bridge using the same set of metrics (Section 5). Related work is summarized in Section 6, and concluding remarks are provided in Section 7.

2. BACKGROUND – GAMESS

The General Atomic and Molecular Electronic Structure System (GAMESS) is a general purpose electronic structure code, with a primary focus on *ab initio* quantum chemistry calculations. GAMESS is used on a wide range of processor technologies all over the world and, as such, presents itself as a useful application to assess the performance and energy consumption aspects of upcoming processor architectures.

The computation parallelization in GAMESS is achieved using the DDI (Distributed Data Interface) library [14, 28]. In GAMESS, for each process that computes chemistry calculations, there is an associated "data server" process that services data requests from the distributed arrays. A 16-core run of GAMESS will, for example, have 8 compute processes and 8 data servers. The following specific GAMESS calculations are considered in this work, using the CCD basis set [10]:

Fragment Molecular Orbital (FMO). In FMO, user specified parameters partition a molecular system, which breaks up the computational workload into chunks which are distributed based on the number of compute nodes available. FMO allows for the use of highly accurate *ab initio* quantum chemistry methods on large molecular systems. For this benchmark, FMO is used to calculate Hartree-Fock (HF) SCF energies. The molecular systems investigated are clusters of 32 and 64 water molecules.

Hartree-Fock, Second Order SCF (HF-SOSCF). Hartree-Fock energies are computed for several benchmark systems. Second order orbital optimizations are performed. Both DISK and DIRECT options are evaluated. For DIRECT runs, integrals are recomputed at each SCF iteration instead of reading them from disk. DISK runs retrieve the stored integrals from disk. Three benchmark molecules are considered for the HF-SOSCF calculations: silatrane ($C_7H_{17}O_4NSi$), nicotine ($C_{10}H_{14}N_2$), and trinitrotoluene (TNT) ($C_7H_5N_3O_6$).

Configuration Interaction with Single Excitations (CIS) A CIS energy computation of the porphin molecule ($C_{20}H_{14}N_4$) is analyzed. CIS is an excited state calculation, for which the computational scaling depends on both system size and the number of excited states calculated. A standard HF calculation is first performed to obtain a reference wavefunction, which is used in a linear combination with configuration state functions representing single electron excitations while neglecting orbital optimization.

Second Order Møller-Plesset Perturbation (MP2), and the Resolution-of-Identity Variant (RI-MP2). MP2/RI-MP2 [17] are second-order perturbation theory energy calculations. In these post Hartree-Fock methods, electron correlation effects are added via a perturbative correction. The MP2 energy calculation is more demanding on both CPU and memory compared to both Hartree-Fock and CIS computations. The RI-MP2 [36] approximation can be as much as 40 times faster than MP2 and is nearly as accurate. Notably, the RI-MP2 approximation employs significant use of matrix-matrix operations in comparison to the domination of vector-vector operations in HF and standard MP2 calculations. The TNT molecule is used in evaluation of the MP2 and RI-MP2 energy calculations.

3. METHODOLOGY

3.1 Platforms

The Moonshot system consists of 128 cores in a multi-node server configuration. The system has 16 nodes (also called cartridges) connected via 10GigE interconnect. Each node has 64 GB memory (DDR3-1600) and one AppliedMicro® X-Gene™ 1 processor, which consists of 8 64-bit ARM cores, each of which is clocked at 2.4 GHz. Each core has 32KB L1 data cache and 32KB L1 instruction cache. Each core pair shares a 256KB L2 cache and all 8 cores in a node share 8MB L3 cache. Each node is configured with a solid state drive (SSD) for storage with 120 GB capacity.

In addition to the scaling runs on the Moonshot with multiple core counts, this paper compares the performance and energy efficiency of the Moonshot system to a dual-socket Intel® Ivy Bridge node with 32 GB memory (DDR3-1333). Each socket consists of 8 cores, each of which is clocked at 2.6 GHz. Each core has 32KB L1 data cache, 32KB L1 instruction cache and 256KB L2 cache. 8 cores in a socket share 20MB L3 cache. The system has traditional disks running at 7200 RPM. It was discovered that the performance of some of the benchmark calculations are highly sensitive to the speed of the storage sub-system, so an SSD with 120 GB capacity was added to the Ivy Bridge system. Unless otherwise noted, all of the relevant results presented in this paper are obtained using the SSD drive.

iperf [12] is used to measure the network bandwidth between nodes on the Moonshot and between sockets on the Ivy Bridge system. The tool saturates the link between the nodes/sockets and determines the maximum bandwidth between nodes/sockets. Inter-node bandwidth on the Moonshot system is 9.89 Gbps (limited by the 10 Gbps switch that connects the nodes of the system) and 30.4 Gbps between sockets on the Ivy Bridge.

3.2 GAMESS Compilation Environment

GAMESS is compiled on the Moonshot system using `gcc-5.1.0` with the flags "`-O2 -mcpu=xgene1 -fno-aggressive-loop-optimizations -march=armv8-a -mtune=xgene1`". Same compiler version and analogous optimization flags ("`-O2 -march=IvyBridge -mtune=intel -fno-aggressive-loop-optimizations` ") are used to compile GAMESS on the Ivy Bridge system. Both systems use *ATLAS* [37] version *3.11.34* for BLAS.

3.3 Power Measurement

In addition to performance at different scales, energy consumption is also of interest. Only the dynamic power draw is measured. To accomplish this, the idle power of each system is measured first. The idle power draw is then subtracted from the total power drawn during application runs. Specific methods used to measure the idle power and the active power for each of the systems are described below.

3.3.1 Moonshot

The Moonshot power measurements rely on the HP iLO (integrated Lights-Out) server management technology [1]. iLO allows total power draw measurement at the chassis level, as well as power draw measurement at the cartridge level. The chassis level power, which can be measured at one-second granularity, includes the power drawn by heavy-duty chassis fans. Rapid changes in the rotational speeds of these fans make the power draw measurement at the chassis level noisy and unreliable. Therefore, power measurements are obtained at the per-cartridge level. The per-cartridge power is measured once every 15 seconds, and GAMESS is executed multiple times for a period of at least 3 minutes to obtain a reliable measurement. Power drawn by the networking components can also be measured using the iLO interface. Networking components draw a constant power of approximately 50 watts.

To calculate dynamic power, the idle power is first measured on each of the Moonshot nodes for a period of 30 minutes. The total dynamic power for a scaling run that uses n nodes is calculated using the following formula:

$$P_{dyn} = (\sum_{i=1}^{n}(L_{p_i} - I_{p_i})) + \frac{N_p}{n} \quad (1)$$

In Equation 1, L_{p_i} is the on-load power drawn by node i, I_{p_i} is the idle power drawn by node i, and N_p is the constant network power. The second term in the equation adds a proportional network power to the dynamic power (P_{dyn}) calculation. Each run produces a series of series of P_{dyn} measurements, which are then averaged to get a power draw value for the run.

3.3.2 Ivy Bridge

Power measurements on the Ivy Bridge are taken using a wattsup [2] device which provides total system power measurement at one second granularity. The device provides a USB interface to obtain the measurements, and this interface is queried from a separate system to ensure that the measurement tools do not increase power draw or slow down the application.

The system fans are set to manual control in the BIOS and their power draw is lumped into the idle power measurement so that their affect on dynamic power draw is minimized. Idle power is measured for 30 minutes and P_{dyn} is calculated by deducting idle power from on-load power.

3.4 Metrics

Power, performance, and energy consumption are measured for each of the GAMESS calculations considered for this work. On the Moonshot system, these calculations were run using 8, 16, 32, 64 and 128 cores; on the Ivy Bridge system, the calculations were run using 8 and 16 cores. For scaling studies on Moonshot, an 8-core single node run is taken as the reference, and all measurements are normalized using the measurements in the reference run. A parallel efficiency metric, defined as:

$$P_{e_n} = \frac{T_1}{n \times T_n} \quad (2)$$

is also measured. In Equation 2, P_{e_n} is the parallel efficiency metric for n nodes (i.e., $n \times 8$ cores), T_1 is the execution time on one node (8 cores) and T_n is the execution time on n nodes. A P_{e_n} metric equal to 1 indicates perfect scaling.

For cross-architecture comparisons, two sets of results are presented—one that takes a single socket 8-core run on the Ivy Bridge system as the reference and normalizes the measurements across different core counts on the Moonshot using the reference measurements, and another that takes the dual-socket 16-core run on the Ivy Bridge system as the reference for normalization.

In addition to power, performance and energy, the Energy-Delay Product (EDP) is calculated as $(energy \times performance)$. The EDP metric emphasizes performance and is widely used in comparing the efficiency of different high-end systems [15].

3.5 Performance Analysis Tools

To analyze performance on the Intel system, a suite of tools developed on top of a binary instrumentation toolkit, PEBIL [23], was used. These tools combine static binary analysis information (e.g., approximate structure of the program in terms of functions and loops, and operations within those structures) with dynamic analyses (e.g., basic block counts and cache simulation) to provide an in-depth description of the performance related characteristics of applications. EPAX [11] was used to analyze the Moonshot system. EPAX provides static binary analyses for both 32-bit and 64-bit ARM architectures. A set of tools that leverages the EPAX static analysis information has also been used to, for example, generate an instruction mix for different structures in the application.

To analyze multi-node runs, the PSiNSTracer tool [35], a light-weight tool that captures communication and computation profiles of MPI applications, was used. The tool intercepts all the MPI calls and collects time spent on each of those calls. The portion of the application execution time that is not attributed to the MPI events is categorized as computation time. This paper analyzes the communication versus computation behavior for only the compute processes. Data servers perform only data-servicing tasks.

4. MOONSHOT PERFORMANCE AND ENERGY

This section describes performance scaling and energy efficiency results obtained on the Moonshot system. Each metric is collected at five core-counts on the Moonshot system: 8, 16, 32, 64 and 128. To reduce noise, each data point is measured five times and the average is reported. For each of the GAMESS calculations considered in the paper, the single-node 8-core run is taken as the reference; measurements across other core counts on the Moonshot systems are normalized using the reference measurements. In Figure 1, the parallel efficiency for each calculation is shown. Values closer to 1 indicate better scaling. In Figure 1 the parallel efficiency for each calculation is shown. Values closer to 1

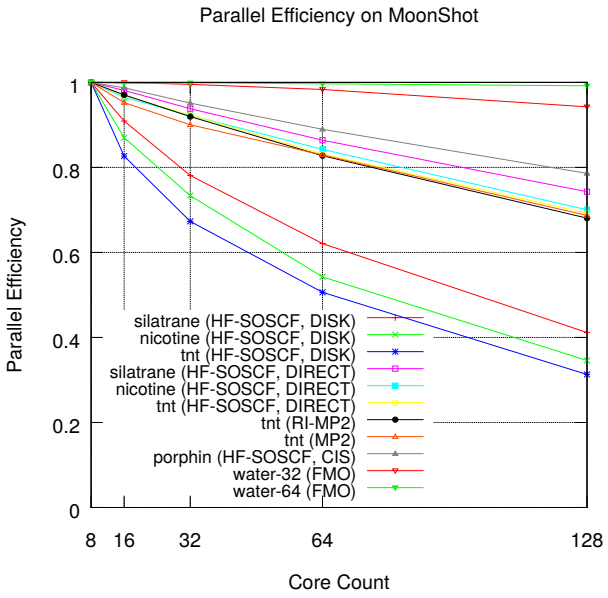


Figure 1: Moonshot Results: Parallel Efficiency (higher is better for large core counts)

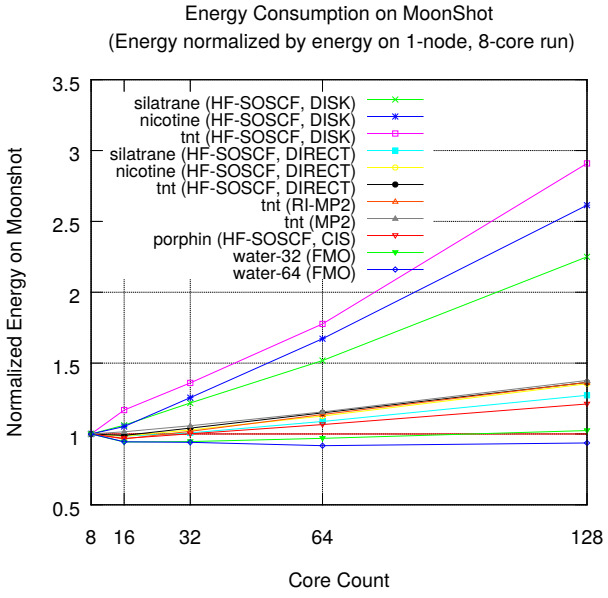


Figure 2: Moonshot Results: Normalized Energy (lower is better for large core counts)

indicate better scaling. In Figure 2, the energy consumption normalized by the single node case is presented. Normalized EDP values are shown in Figure 3. Lower values indicate greater efficiency.

4.1 FMO Results

FMO calculations show near linear scaling on Moonshot. For a 128-core run with 64 water molecules, the speedup with respect to the 8-core run is 15.9 (linear speedup would

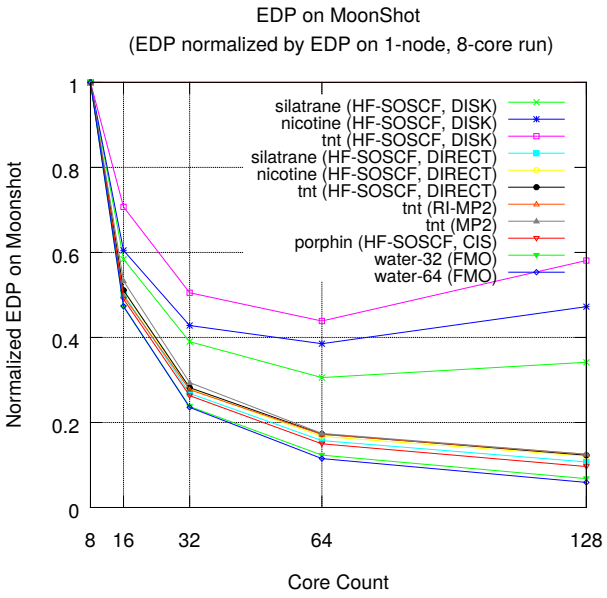


Figure 3: Moonshot Results: Normalized EDP (lower is better for large core counts)

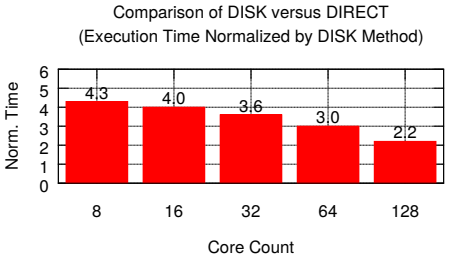


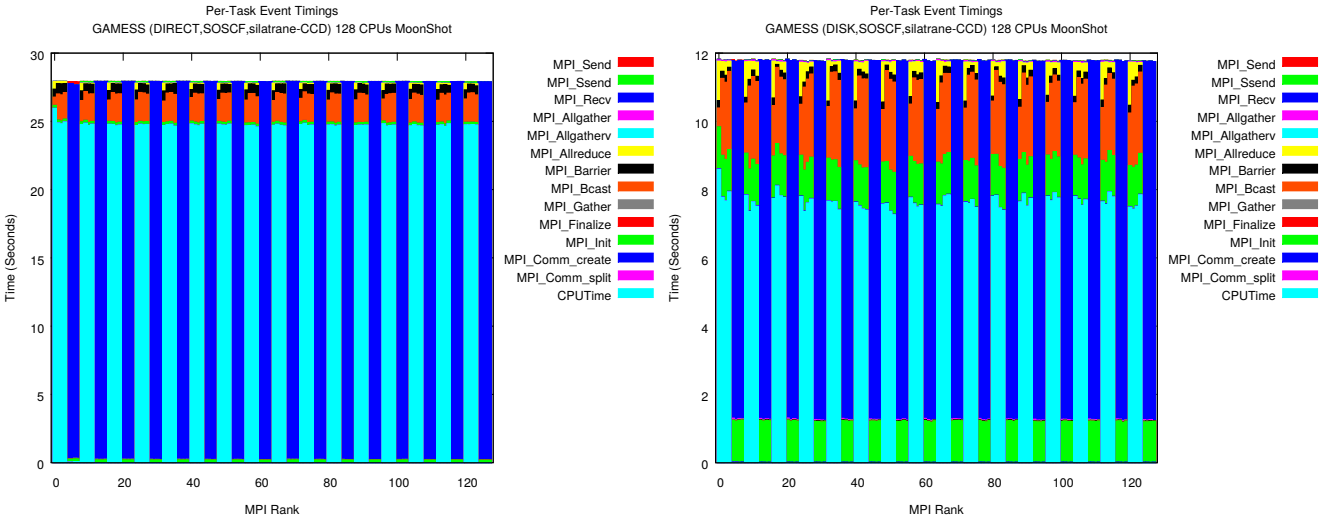
Figure 4: Comparing the performance of DISK and DIRECT methods across multiple core-counts on the Moonshot system.

be 16). As mentioned earlier, the implementation of FMO breaks the computational workload into chunks, which are then distributed to available compute nodes. The communication between the nodes is minimal, which explains the scaling behavior. Communication-computation profiles reveal that for 128-core runs with 64 waters, on average, less than 5% of the time spent by compute processes is attributed to communication. Normalized EDP (as shown in Figure 3) improves the most for FMO calculations at higher core counts on the Moonshot system compared to other calculations. Results for the 32 water system are similar to those for the 64 water system.

4.2 HF-SOSCF Results

Recall from Section 2 that both disk and direct variants of HF-SOSCF calculations are considered. The relative performance of the two methods (disk versus direct) was investigated using the silatrane molecule, in addition to analysis of the performance scaling, parallel efficiency and EDP.

The normalized execution times for the direct method HF-SOSCF calculations are shown in Figure 4. The execution



(a) MPI profile for 128-core run of silatrane on Moonshot using direct method.

(b) MPI profile for 128-core run of silatrane on Moonshot using disk method.

Figure 5: MPI profiles for silatrane (HF-SOSCF) molecule on Moonshot.

times for the direct method are normalized to the execution times for the disk method. As demonstrated in the graph, disk based calculations are faster at all core counts. However, the edge that disk based calculations has on direct calculations at 8-core runs diminishes significantly at 128-core runs.

Next, the parallel efficiency metric is considered. It is shown in Figure 1 that disk based methods rank lowest in terms of parallel efficiency. Parallel efficiency for the silatrane molecule using disk is at 0.4 (at 128 cores) versus 0.74 for the direct method[1]. Normalized EDP points to the same conclusion. The normalized EDP for disk based calculations stops declining beyond 64-core count runs (Figure 3); i.e., efficiency stops improving beyond 64 cores.

To further investigate the relative difference in scaling between the direct and disk based methods, consider the communication and computation profiles for these methods. Profiles for both the direct and disk methods for the silatrane molecule calculations executed using all 128 cores (16 nodes) on the Moonshot system are shown in Figure 5. Poor parallel efficiency with the disk based method can be primarily attributed to the greater communication intensity of this method compared to the direct method. For 128 core direct runs, time spent in communication by compute processes of GAMESS is 11%, compared to 35% for disk runs. Most of this communication time is spent in MPI_Bcast events.

4.3 CIS Results

The HF Configuration Interaction-Singles (CIS) calculation using the porphin molecule ranks second after the FMO method in parallel efficiency (0.79, see Figure 1). The normalized EDP metric indicates higher efficiency at higher core count runs. On average, GAMESS compute processes are engaged in communication events for 13% of the run time for 128-core runs.

[1]Parallel efficiency quantifies what proportion of theoretical maximum speedup is achieved by GAMESS when scaled to multiple nodes on the Moonshot. 0.74 means 74% of the achievable speedup was attained.

4.4 MP2/RI-MP2 Results

These calculations show greater parallel efficiency than HF disk based methods but lower efficiency than FMO calculations. Both MP2 and RI-MP2 show identical parallel efficiencies of 0.68 for 128 core runs (Figure 1). The normalized EDP metric, which emphasizes performance, continues to decline (Figure 3) for large core counts (i.e., strong scaling will continue beyond 128-core runs). Analyses of the computation-communication profile for 128-core runs reveal that compute processes in MP2 calculations, on average, are engaged in communication events for 19% of the run time, compared to 20% of RI-MP2.

5. CROSS-ARCHITECTURAL STUDY

To make cross-architectural performance and energy comparisons, the playing field has been made as level as practicably possible, by using the exact same compilation environment on the two systems. In the preliminary analysis of the performance differences between the two systems, it was discovered that the Moonshot system performed relatively better (even for the same core-counts) than the Ivy Bridge system for HF-SOSCF disk-based calculations. An I/O profiling analysis (performed with the I/O tracer built on top of PEBIL [27]) revealed that the Ivy Bridge system, which was utilizing the traditional spinning hard-disk drive, was spending considerable time on I/O calls. An SSD with similar performance specifications to that of the SSD on the Moonshot was added to the Ivy Bridge system, and that addition improved the performance of HF-SOSCF disk based calculations by up to 2.2×. This paper reports the performance for HF disk based calculations using the SSD on the Ivy Bridge system.

The performance, energy and EDP metrics obtained on the Moonshot system across multiple core counts were normalized to the same metrics obtained on single-socket (8-core) and dual-socket (16-core) runs on the Ivy Bridge

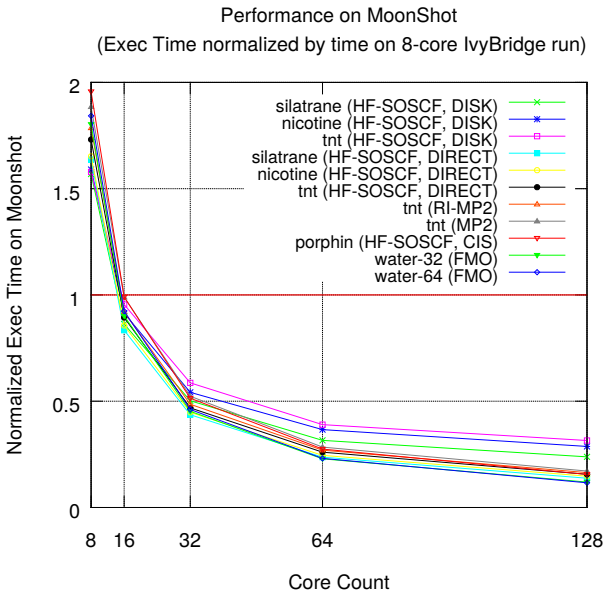


Figure 6: Cross Architecture Comparison: Performance on Moonshot normalized by performance on 8-core (one-socket) of Ivy Bridge (lower is better for Moonshot).

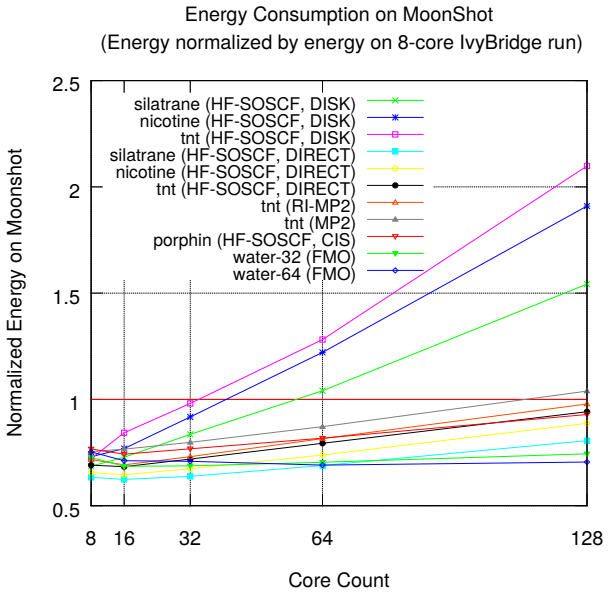


Figure 7: Cross Architecture Comparison: Energy consumed on Moonshot normalized by the energy on 8-core (one-socket) of Ivy Bridge (lower is better for Moonshot).

system. These comparisons facilitate the investigation of whether using more low-power simpler cores can be beneficial in terms of energy efficiency compared to using relatively few heavy-duty cores.

5.1 Single Socket Ivy Bridge

Comparative results are shown in Figures 6, 7, and 8 which

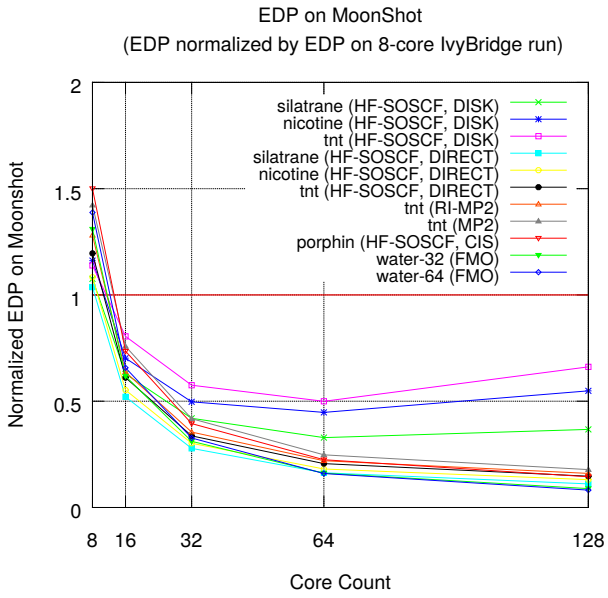


Figure 8: Cross Architecture Comparison: EDP or efficiency on Moonshot normalized by EDP on 8-core (one-socket) of Ivy Bridge (lower is better for Moonshot).

use 8-core Ivy Bridge metrics as a reference. In Figure 6, the normalized performance across multiple core count runs on the Moonshot system is shown. Performance on the Moonshot using 8 cores is 1.6× to 2× slower than the 8 core runs on Ivy Bridge. All 8-core runs on the Moonshot use less energy than the corresponding 8-core runs on Ivy Bridge (63% to 77% of the energy needed to run on Ivy Bridge). The normalized EDP metric for 8-core runs, however, show that the Ivy Bridge system is more efficient at running 8-core executions. As the Moonshot scales to 16 cores, the performance in all cases exceeds the performance on 8 cores of the Ivy Bridge. The normalized EDP metric also suggests that greater efficiency can be achieved with large core count runs on the Moonshot. The next natural research questions to ask are—What drives the performance differences between 8-core runs on Moonshot and Ivy Bridge, and what architectural components on the Moonshot tend to bottleneck its performance?

Architectural Bottlenecks. The Moonshot system exposes a set of performance hardware counters that can be used to measure the interactions of software with key on-node architectural components of Moonshot—floating point units, caches, memory, and the branch predictor. These counters can be measured using PAPI [25], which has support for the X-Gene 1 architecture. The overall idea is to investigate which of these counters correlate with the relative performance difference between 8-core runs on Moonshot and Ivy Bridge. Clearly, measuring these counters for just the 11 GAMESS calculations considered in this paper may be an insufficient number of data points for correlation analysis. Therefore, a set of HPC application benchmarks from different scientific domains is added to the analysis.

This extended set of benchmarks includes: a subset of calculations from the NAS Parallel Benchmarks

(CG, FT, IS, LU and MG) [5], CoMD (molecular dynamics) [3], lulesh (shock hydrodynamics) [21], miniFE (finite element) [19], miniGhost (finite difference) [6], and smg2000 (semi-coarsening multigrid) [7]. These applications are run with different input sets to generate a total of 38 data-points to supplement the 11 data-points from GAMESS. Each data-point consists of 30 performance hardware counters that measure the number of total instructions executed, the number of floating point instructions, the number of loads/stores from L1 data cache, the number of branch instructions, the number of branches accurately predicted, etc.

In the correlation analysis, all counters are first normalized for a given application by the number of total instructions executed by the application. The correlation coefficients are then calculated for each of the normalized counters, and the ratio of performance on the 8-core Moonshot to that on the 8-core Ivy Bridge. Only the counters that have absolute correlation coefficients of more than 0.6 are considered[2] .

Floating Point/Integer Performance: Counters that measure floating point operations and integer operations rank among the highest: 0.7 correlation coefficient for floating point operations and 0.61 for integers. The number of floating point operations per instruction is positively correlated to the relative performance, suggesting that floating point heavy calculations perform slower on the Moonshot, while the number of integer operations is negatively correlated to the relative performance. The floating point correlation could be attributed to 1) the faster CPU clock on the Ivy Bridge (2.6GHz versus 2.4GHz on Moonshot), and 2) different theoretical floating point operations per cycle for the two systems: 16 single precision flops per cycle (using 8 fused multiply-adds) on Ivy Bridge compared to 8 single precision flops per cycle (using 4 fused multiply-adds) on the Moonshot.

Memory Subsystem Performance: Performance counters which measure the interactions of applications with the memory subsystem also register high correlations. In particular, data load instructions are correlated to the relative performance with a coefficient of 0.6; higher data loads per instruction lead to lower relative performance on the Moonshot system. Cache and main memory read bandwidths are measured on both systems using the *lmbench* tool [26]. Per core main memory read bandwidth on the Ivy Bridge system is 1.45× greater than that on the Moonshot system. L1-cache read bandwidth is 2× higher on the Ivy Bridge system.

Branch Predictor Performance: Performance counters that measure branches show high correlation with relative performance, with a coefficient of 0.6. The relative performance of the Moonshot improves as more branches are accurately predicted. This suggests differences in the capabilities of the branch prediction units on the two systems. There appear to be no previous accurate and verifiable studies that look at the branch unit performance. This paper introduces a benchmark intended to measure the cost of mis-predicted branches.

[2]Correlation coefficients range from -1 to +1, where -1 indicates perfect negative correlation and +1 indicates perfect positive correlation

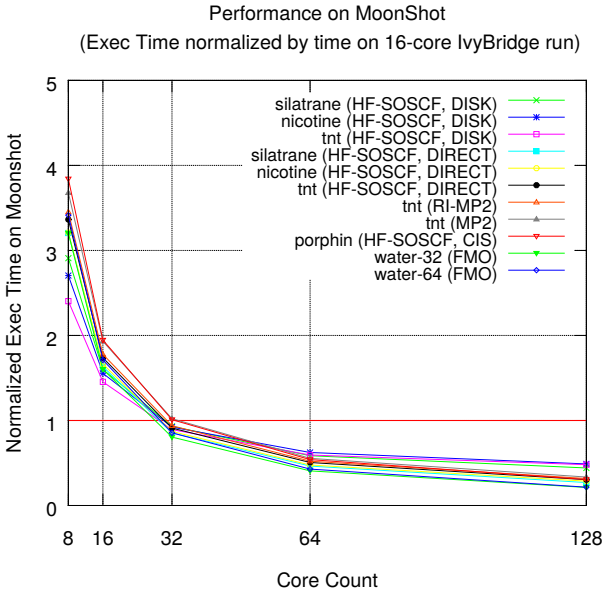


Figure 9: Cross Architecture Comparison: Performance on Moonshot normalized by performance of 16-core runs on Ivy Bridge (lower is better for Moonshot).

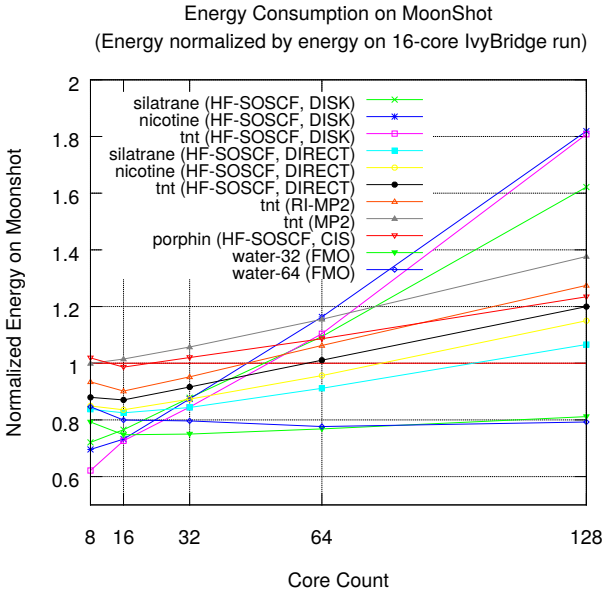


Figure 10: Cross Architecture Comparison: Energy consumed on Moonshot normalized by the energy of 16-core runs on Ivy Bridge (lower is better for Moonshot).

The benchmark consists of a small loop containing a single branch. Each path of the branch increments a counter to prevent the path from being optimized away. The direction of the branch is determined by the value of a byte read from an array, either 0 or 1. The array is indexed by the loop iteration modulo the size of the array. Each entry in the array is initialized by rand () modulo 2. The configurable param-

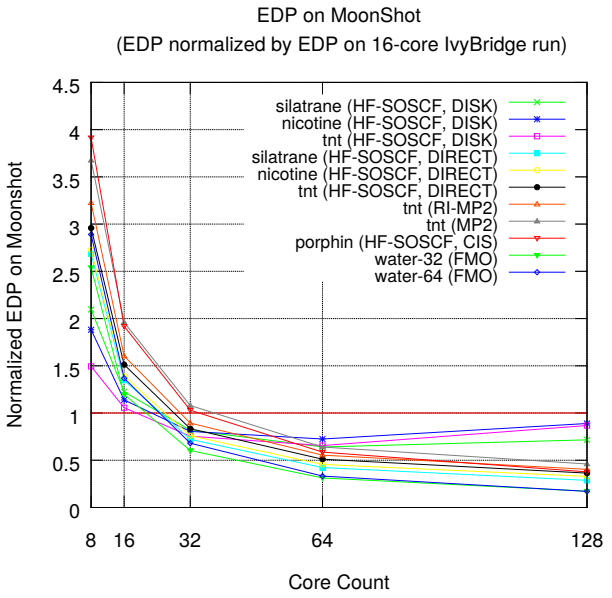


Figure 11: Cross Architecture Comparison: EDP or efficiency on Moonshot normalized by EDP of 16-core runs on Ivy Bridge (lower is better for Moonshot).

eters for the benchmark are the number of loop iterations and the size of the array.

The loop is optimized similarly on both the Ivy Bridge and Moonshot systems. The loop consists of 3 blocks of 4 instructions each. A head block reads the branch direction from the array and branches to one of the other two blocks. Each of the other two blocks increment the loop counter, increment the path counter, and branches either to the loop head or departs the loop. So, each iteration of the loop consists of 8 instructions, 2 of which are branches. One of these branches is a loop branch, which should almost always be correctly predicted, and the other is a path branch, which should have a 50% mis-prediction rate. Four hardware counters are collected for several configurations of the benchmark: total instructions, total cycles, branch instructions (*br_ins*) and mis-predicted branches (*br_msp*).

Verification of whether the benchmark causes branch mispredictions can be done by comparing *br_msp* to the number of loop iterations. If the benchmark is sufficient to break the branch predictor, then there should be approximately 1 mis-prediction per 2 iterations. An array of 32768 random branch directions is sufficient to obtain 0.50 branch mispredictions per iteration on the Moonshot, but only 0.46 branch mis-predictions per iteration on the Ivy Bridge. An array of 65536 entries achieved 0.49 mis-predictions per iteration on the Ivy Bridge. This indicates that the Ivy Bridge is more capable of predicting branch directions for this benchmark than the Moonshot.

To measure the cycle impact of each mis-prediction, the benchmark is modified so that every entry in the array is 0. This allows the branch predictors on both systems to nearly always predict the correct branch path. The cost of a mispredicted branch is then quantified as the difference in total cycles between the two benchmarks per *br_msp*. A branch mis-prediction on the Ivy Bridge increases the total cycle count by 23 cycles while a mis-prediction on the Moonshot increases the total cycle count by 26 cycles. As the Ivy Bridge has a pipeline of only 14-19 cycles, this also suggests that an application may experience performance loss on a branch mis-prediction in excess of the 14 cycles it takes to flush the pipeline.

Relationship of Findings to GAMESS Calculations: To put all of the bottleneck analysis discussion into perspective for the GAMESS calculations studied in this paper, consider two calculations that show the highest and the lowest relative performance. The research question is whether the relevant performance counters for those calculations corroborate our findings. The HF-SOSCF CIS calculation using the porphin molecule run on 8 cores is 2× slower on the Moonshot than on Ivy Bridge. The HF-SOSCF CCD calculation done using the disk based method using the silatrane molecule is 1.6× slower on the Moonshot system. In terms of the floating point operations, the porphin calculation has 461 floating point operations for every 1000 instructions, while the silatrane calculation has 390 floating point instructions for every 1000 instructions. In terms of the load operations, the porphin calculation has 274 load operations every 1000 instructions while the silatrane disk based calculation has 257. Finally, branches are predicted with 97% accuracy[3] for the porphin molecule and with 98% accuracy for the silatrane disk based calculation.

5.2 Dual Socket Ivy Bridge

Figures 9, 10, 11 show the cross-architecture comparison results that use 16-core Ivy Bridge runs as the reference. In the Figure, the performance, energy, and EDP of executions performed across multiple core counts on the Moonshot system are normalized using the corresponding reference metric on 16-core Ivy Bridge executions. For a given calculation and core-count run on Moonshot, a value of less than 1 for performance, energy or EDP (Figures 9, 10, 11) indicates the performance (energy or EDP) on the Moonshot is better than 16-core run of the same calculation on the Ivy Bridge system. The key conclusion here is that the performance of the GAMESS calculations using 16 cores with Ivy Bridge is matched by a 32-core run on the Moonshot system, suggesting a 2:1 ratio for Moonshot core to Ivy Bridge core performance. However, as mentioned in Section 3.1, Ivy Bridge dual socket executions get up to 3 times more communication bandwidth (because the communication is inter-socket) than the multi-node executions on the Moonshot (32 core runs use 4 nodes).

It was noted in Section 4.2 that HF-SOSCF calculations which utilize disk to store and retrieve the integrals are engaged in communication events for a significant portion of the total run time. Further analysis of the computation-communication profiles of this case reveals that the 16-core run of a silatrane disk-based calculation on the Moonshot are in communication events for 1.4× more time than the same core count run on the Ivy Bridge. Therefore, the 2:1 core-to-core Moonshot:Ivy Bridge ratio is the upper bound when it comes to the performance of GAMESS calculations considered in this paper. This ratio should improve in favor of ARMv8 with improvements in the interconnect technology.

[3]We define accuracy as: $(br\_ins\text{-}br\_msp)/br\_ins \times 100$

6. RELATED WORK

The potential of the ARM architecture for energy efficient HPC has been recognized by both system vendors and the academic research community. Vendors such as Cavium [8] and HP [18] have started to bring new products to market with the HPC community in mind, as well as researchers reporting on their own findings of the ARM as a potential design point in HPC servers [9, 34]. The Barcelona Supercomputing Center designed a testbed cluster from ARMv7 processors and a 1GbE network and stressed the need for an optimized software stack and a high performance network [31].

Performance engineers have been reporting their findings on the ARM performance on scientific codes as well [4,29,33]. Padoin et al. report the energy efficiency and performance of an ARMv7 processor on the NAS Parallel Benchmarks compared to a Sandy Bridge processor. Their findings show that while ARM uses less power, it is less energy efficient than the Sandy Bridge due to its lower performance.

Laurenzano et al. [24] report the effectiveness of the ARMv7 on HPC computational benchmarks. The authors concluded that ARMv7 FP/SIMD and memory subsystem performance would need to be improved in order to be a viable option for use by the scientific community.

Multiple papers have focused on GAMESS performance on different offerings of HPC systems [13, 16]. A recent study by Keipert et al. [22] compared intra-node performance and energy efficiency of GAMESS on commercially available x86, 32-bit ARM, and 64-bit ARM systems. Jundt et al. [20] adopt a machine-learning based methodology to learn the on-node architectural bottlenecks on 64-bit ARM system. The present work is the first study that takes a comprehensive look at the inter-node performance, parallel efficiency and energy efficiency of different types of GAMESS calculations on a commercially available 64-bit ARM cluster.

7. DISCUSSION AND CONCLUSIONS

This paper presented an analysis of the performance, parallel scalability and energy efficiency of a widely used quantum chemistry code, GAMESS, on a commercially available HP Moonshot 64-bit ARM cluster. GAMESS calculations explored in this work scale to larger core counts on the Moonshot system; the extent of the speedup is different for different types of calculations. In terms of the EDP metric, higher core count runs on Moonshot are almost always more efficient than lower count runs. These results show great promise for the co-design approach that considers using many low-power cores rather than a relatively few heavy-duty powerful cores to deliver an Exaflop system that can operate within in the 20MW power envelope.

A cross-architecture comparison of performance and energy efficiency metrics was also presented, based on the Intel Ivy Bridge system as the reference. For most of the benchmarking inputs used in the study, the performance on one node of Ivy Bridge with 16 cores is matched by a four node run (with 32 cores) on the Moonshot, notwithstanding the fact that 16-core runs on Ivy Bridge benefit from higher inter-socket communication bandwidth than the inter-node runs on the 64-bit ARM cluster. The results are interesting and encouraging given the relatively nascent entrance of ARM into the HPC world. Advancements in the compiler and the software stacks for the 64-bit ARM architecture, which have only had a short time to evolve, will mitigate some of the performance bottlenecks in the 64-bit ARM architecture.

Acknowledgements

This work was supported in part by the Air Force Office of Scientific Research under AFOSR Award No. FA9550-12-1-0476 and by DoE SBIR Award No. DE-SC0013164. Sponsorship of the Department of Defense High Performance Computing Modernization Program, through a HASI grant, is gratefully acknowledged.

8. REFERENCES

[1] Server remote management with HP Integrated Lights Out (iLO). `http://tinyurl.com/o6so5bk`.

[2] WattsUp? Meters. `https://www.wattsupmeters.com/`.

[3] CoMD Proxy Application. `http://www.exmatex.org/comd.html`, 2015.

[4] D. Abdurachmanov, B. Bockelman, P. Elmer, G. Eulisse, R. Knight, and S. Muzaffar. Heterogeneous high throughput scientific computing with apm x-gene and intel xeon phi. *Journal of Physics: Conference Series*, 608(1):012033, 2015.

[5] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The nas parallel benchmarks - summary and preliminary results. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, Supercomputing '91, pages 158–165, New York, NY, USA, 1991. ACM.

[6] R. F. Barrett, C. T. Vaughan, and M. A. Heroux. Minighost: a miniapp for exploring boundary exchange strategies using stencil computations in scientific parallel computing. *Sandia National Laboratories, Tech. Rep. SAND*, 5294832, 2011.

[7] P. N. Brown, R. D. Falgout, and J. E. Jones. Semicoarsening Multigrid on Distributed Memory Machines. *SIAM J. Sci. Comput.*, 21(5):1823–1834, 2000.

[8] Cavium. ThunderX ARM Processors. `http://tinyurl.com/mj2ayo4`, 2015.

[9] M. F. Cloutier, C. Paradis, and V. M. Weaver. Design and analysis of a 32-bit embedded high-performance cluster optimized for energy and performance. In *Proceedings of the 1st International Workshop on Hardware-Software Co-Design for High Performance Computing*, Co-HPC '14, pages 1–8, Piscataway, NJ, USA, 2014. IEEE Press.

[10] T. H. Dunning. Gaussian basis sets for use in correlated molecular calculations. i. the atoms boron through neon and hydrogen. *The Journal of Chemical Physics*, 90(2):1007–1023, 1989.

[11] EP Analytics, Inc. EPAX Toolkit: Binary Analysis for ARM. `http://epaxtoolkit.com`, 2014.

[12] J. D. et al. iPerf – The network bandwidth measurement tool. `https://iperf.fr/`, 2015.

[13] G. D. Fletcher, D. G. Fedorov, S. R. Pruitt, T. L. Windus, and M. S. Gordon. Large-scale mp2

calculations on the blue gene architecture using the fragment molecular orbital method. *Journal of Chemical Theory and Computation*, 8(1):75–79, 2012.

[14] G. D. Fletcher, M. W. Schmidt, B. M. Bode, and M. S. Gordon. The distributed data interface in gamess. *Computer Physics Communications*, 128:190 – 200, 2000.

[15] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors, 1996.

[16] M. S. Gordon, D. G. Fedorov, S. R. Pruitt, and L. V. Slipchenko. Fragmentation methods: A route to accurate calculations on large systems. *Chemical Reviews*, 112(1):632–672, 2012. PMID: 21866983.

[17] M. Head-Gordon, J. A. Pople, and M. J. Frisch. Mp2 energy evaluation by direct methods. *Chemical Physics Letters*, 153(6):503 – 506, 1988.

[18] N. Hemsoth. Moonshot Moves HPC Closer to ARM's Reach. `http://tinyurl.com/pvuwnpb`, 2015.

[19] M. A. Heroux, D. W. Doerfler, P. S. Crozier, J. M. Willenbring, H. C. Edwards, A. Williams, M. Rajan, E. R. Keiter, H. K. Thornquist, and R. W. Numrich. Improving performance via mini-applications. *Sandia National Laboratories, Tech. Rep. SAND2009-5574*, 2009.

[20] A. Jundt, A. Cauble-Chantrenne, A. Tiwari, J. Peraza, M. Laurenzano, and L. Carrington. Compute bottlenecks on the new 64-bit arm. In *International Workshop on Energy Efficient Supercomputing (E2SC)*, E2SC '15. IEEE, 2015. To Appear.

[21] I. Karlin, A. Bhatele, J. Keasler, B. Chamberlain, J. Cohen, Z. DeVito, R. Haque, D. Laney, E. Luke, F. Wang, et al. Exploring traditional and emerging parallel programming models using a proxy application. In *Proceedings of the 2013 IEEE International Symposium on Parallel and Distributed Processing*, IPDPS, 2013.

[22] K. Keipert, G. Mitra, V. Sundriyal, S. S. Leang, M. Sosonkina, A. P. Rendell, and M. S. Gordon. Energy efficient computational chemistry: Comparison of x86 and arm systems. *Journal of Chemical Theory and Computation*, 2015. `http://dx.doi.org/10.1021/acs.jctc.5b00713`, Articles ASAP.

[23] M. Laurenzano, M. Tikir, L. Carrington, and A. Snavely. Pebil: Efficient static binary instrumentation for linux. In *Performance Analysis of Systems Software (ISPASS), 2010 IEEE International Symposium on*, pages 175 –183, march 2010.

[24] M. Laurenzano, A. Tiwari, A. Jundt, J. Peraza, J. Ward, WilliamA., R. Campbell, and L. Carrington. Characterizing the performance-energy tradeoff of small arm cores in hpc computation. In *Euro-Par 2014 Parallel Processing*, volume 8632, pages 124–137. Springer International Publishing, 2014.

[25] K. London, S. Moore, P. Mucci, K. Seymour, and R. Luczak. The papi cross-platform interface to hardware performance counters. In *Department of Defense Users' Group Conference Proceedings*, pages 18–21, 2001.

[26] L. McVoy and C. Staelin. lmbench: portable tools for performance analysis. In *Proceedings of the 1996 annual conference on USENIX Annual Technical Conference*, ATEC '96, pages 23–23, Berkeley, CA, USA, 1996. USENIX Association.

[27] M. Meswani, M. Laurenzano, L. Carrington, and A. Snavely. Modeling and predicting disk i/o time of hpc applications. In *High Performance Computing Modernization Program Users Group Conference (HPCMP-UGC), 2010 DoD*, pages 478–486, June 2010.

[28] R. Olson, M. Schmidt, M. Gordon, and A. Rendell. Enabling the efficient use of smp clusters: The gamess/ddi model. In *Supercomputing, 2003 ACM/IEEE Conference*, pages 41–41, Nov 2003.

[29] E. L. Padoin, L. L. Pilla, M. Castro, F. Z. Boito, P. O. A. Navaux, and J.-F. Méhaut. Performance/energy trade-off in scientific computing: the case of arm big. little and intel sandy bridge. *IET Computers & Digital Techniques (CDT)*, 2014.

[30] N. Rajovic, N. Puzovic, L. Vilanova, C. Villavieja, and A. Ramirez. The low-power architecture approach towards exascale computing. In *Proceedings of the Second Workshop on Scalable Algorithms for Large-scale Systems*, ScalA '11, pages 1–2, New York, NY, USA, 2011. ACM.

[31] N. Rajovic, A. Rico, N. Puzovic, C. Adeniyi-Jones, and A. Ramirez. Tibidabo1: Making the case for an arm-based {HPC} system. *Future Generation Computer Systems*, 36:322 – 334, 2014.

[32] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, and J. A. Montgomery. General atomic and molecular electronic structure system. *Journal of Computational Chemistry*, 14(11):1347–1363, 1993.

[33] L. Stanisic, B. Videau, J. Cronsioe, A. Degomme, V. Marangozova-Martin, A. Legrand, and J.-F. Mehaut. Performance analysis of hpc applications on low-power embedded platforms. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013.

[34] P. Stanley-Marbell and V. Cabezas. Performance, power, and thermal analysis of low-power processors for scale-out systems. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 863–870, May 2011.

[35] M. Tikir, M. Laurenzano, L. Carrington, and A. Snavely. Psins: An open source event tracer and execution simulator for mpi applications. In H. Sips, D. Epema, and H.-X. Lin, editors, *Euro-Par 2009 Parallel Processing*, volume 5704 of *Lecture Notes in Computer Science*, pages 135–148. Springer Berlin Heidelberg, 2009.

[36] F. Weigend, M. HÃd'ser, H. Patzelt, and R. Ahlrichs. Ri-mp2: optimized auxiliary basis sets and demonstration of efficiency. *Chemical Physics Letters*, 294:143 – 152, 1998.

[37] R. C. Whaley and J. J. Dongarra. Automatically tuned linear algebra software. In *Proceedings of the 1998 ACM/IEEE Conference on Supercomputing*, SC '98, pages 1–27, Washington, DC, USA, 1998. IEEE Computer Society.