# Power and Performance Analysis of the Graph 500 Benchmark on the Single-chip Cloud Computer

Zhiquan Lai*, King Tin Lam†, Cho-Li Wang†, Jinshu Su‡*
*College of Computer, National University of Defense Technology, Changsha, China
†Department of Computer Science, The University of Hong Kong, Hong Kong, China
‡National Key Laboratory of Parallel and Distributed Processing (PDL), Changsha, China
{zqlai, sjs}@nudt.edu.cn, {ktlam, clwang}@cs.hku.hk

***Abstract*—Data-intensive applications are playing increasingly important role in HPC systems and data centers. Graph 500 benchmark (Graph500) is a widely used data-intensive benchmark for evaluating the performance of computing systems. On the other hand, Energy efficiency is quickly becoming a first-order design constraint of these systems. In this paper, we introduce a data-level paralleled implementation of Graph500 on a typical DVFS-featured many-core architecture, Intel Single-chip Cloud Computer (SCC). The parallelism, which follows a shared virtual programming model, is exploited by the shared physical memory (SPM) on SCC. We thoroughly evaluate the power and performance characteristics of Graph500 on various scale of cores at different power states. The experimental results provide valuable insight for designing an energy-efficient many-core system for data-intensive applications.**

***Keywords*—power; performance; Graph 500; many-core; Intel SCC**

## I. Introduction

In the past few years, data-intensive applications are playing increasingly important role in HPC systems and data centers. Graph 500 benchmark (Graph500) [1], which performs breadth-first search (BFS) on a large-scale graph, is a widely used data-intensive benchmark for evaluating the performance of computing systems. There is rich literature on parallelization (and optimization) for Graph500 [2]–[4]. On the other hand, Energy efficiency is quickly becoming a first-order design constraint of HPC systems and data centers. The top-ranked supercomputer, China's Tianhe-2 [5], consumes 17.81 MW (excluding the cooling system), which is equivalent to the domestic power consumption of a midsize town (over 312,800 persons) in China. Energy consumption is as important as system performance [6]. Dynamic voltage and frequency scaling (DVFS) is an widely used technique to achieve a power-performance tradeoff.

Many-core architectures boost the performance of the HPC systems, meanwhile consuming the great portion of the system power. Intel Single-chip Cloud Computer (SCC) [7] is a typical DVFS-featured many-core architecture. However, as we known, SCC still has not been fully evaluated for power and performance of Graph500.

In this paper, we introduce a data-level paralleled implementation of Graph500 on SCC. The parallelism, which follows a shared virtual programming model, is exploited by the shared physical memory (SPM) on SCC. We thoroughly evaluate the power and performance characteristics of Graph500 on various scale of cores at different power states. The experimental results provide some useful hints for energy-efficient programming and system design.

The remainder of this paper is organized as follows. Section II describes the parallel Graph500 on SCC. Section III discusses the evaluation methodology. The power and performance characteristics are presented in Section IV. Finally, we conclude the paper in Section V.

## II. Graph500 on Intel SCC

### A. Graph500

Researchers observed that data-intensive supercomputing applications are of growing importance to representing current HPC workloads, but existing benchmarks did not provide useful information for evaluating supercomputing systems for data-intensive applications. In order to guide the design of hardware architectures and software systems to support data-intensive applications, the Graph500 was proposed and developed [1]. The workflow of Graph500 is described in Algorithm 1. Its kernel workload is performing breadth-first searches (BFSes) over a large-scale graph.

In the original Graph500, only step 2 and step 4.2 (i.e. kernel 1 and kernel 2) are timed and included in the performance information. However, the whole benchmark could also be considered as the representative application. Thus, we inspect the power and performance characteristics of both the whole benchmark and the two kernels.

### B. SPM-based Implementation on SCC

Intel SCC [7] is a many-core architecture integrated 48 P54C CPU cores. Each core has local 16KB L1 cache. However, Intel SCC does not provide hardware cache coherence among the 48 cores. Shared physical memory (SPM) is a coherence mechanism managed by software. SPM is a centralized approach bearing much similarity to traditional shared-memory systems (SMP or multi-core) except that the

**Algorithm 1:** Algorithm of Graph500

**Input**:
$SCALE$: the vertices scale, $2^{SCALE}$ vertices
$EDGE$: the edge factor, $EDGE \cdot 2^{SCALE}$ edges

1 **begin**
2  Step 1: Generate the edge list.
3  Step 2: Construct a graph from the edge list. **kernel 1**
4  Step 3: Randomly sample 64 unique search keys.
5  Step 4: **for** *each search key* **do**
6  Step 4.1: Compute the parent array. **kernel 2**
7  Step 4.2: Verify the parent array.
8  Step 5: Compute and output performance information.

cache coherence of the shared memory now becomes software-managed. SPM-mapped shared pages are located in the shared DRAM of SCC and synchronized based on release consistency. We have featured SPM mode in our Rhymes (Runtime with HYbrid MEmory Sharing) [8] library on Barrelfish operating system.

The Graph500 is ported into Rhymes environment and set as SPM mode. The parallel programming model exploits the data-level parallelism and follows the traditional SPMD pattern (single program, multiple data). The master core is responsible for dividing the task to other slave cores and collect the results from slave cores.

## III. Evaluation Methodology

The problem size of Graph500 is set as follows: $SCALE$ = 18 (262,144 vertices) and $EDGE$ factor = 16 (4,194,304 edges). We use gcc version 4.4.3 to compile programs with the O3 optimization level. During the experiments, the clock frequencies of the network-on-chip (NoC) and memory controllers (MCs) are both fixed at 800MHz.

To avoid overhead introduced by power measurement, we measure the power consumption of the SCC outside the chip. As shown in Fig. 1, we view the chip as a "black box" and read the current and voltage values from the board management controller (BMC) using the management console PC (MCPC) during the application run. As shown in Fig. 1, the BMC is connected via Ethernet to the MCPC, from which the power values are read. PCIE bus connects the SCC to the MCPC. The power states of the chip will be recorded into a log file on the MCPC. Although the maximum power sampling rate is only about 3.3 samples per second, it suffices for our evaluation. The average power of an execution is estimated by taking arithmetic mean of all sampled power values.

The CPU cores of SCC are set up using DVFS mechanism at four different power states (voltage/frequency setting), 800MHz/1.1V, 533MHz/0.9V, 400MHz/0.8V and 320MHz/0.8V. For each power state, we launch the benchmark on 1, 2, 4, 8, 16, 32 and 48 cores to inspect the power and performance characteristics.

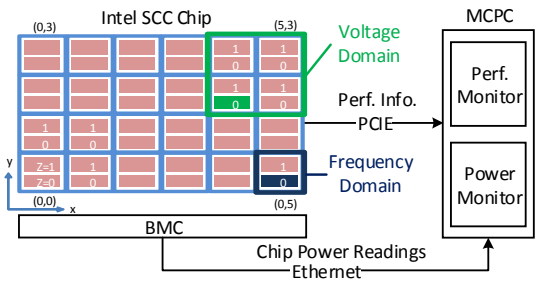



Fig. 1. Power domains of Intel SCC and power measurement setting

## IV. Experimental Results and Analysis

As an example, Fig. 2 presents the instantaneous power when Graph500 executes on 48 cores at 800MHz/1.1V. $P_{core}$ denotes the power of CPU cores and network on-chip, while $P_{ddr}$ for DDR3 memory and $P_{mc}$ for four memory controllers. Then $P$ is the sum of $P_{core}$, $P_{ddr}$ and $P_{mc}$. From the figure, we can find that the $P_{core}$ and $P_{ddr}$ is dynamically changing, while $P_{mc}$ is nearly static.

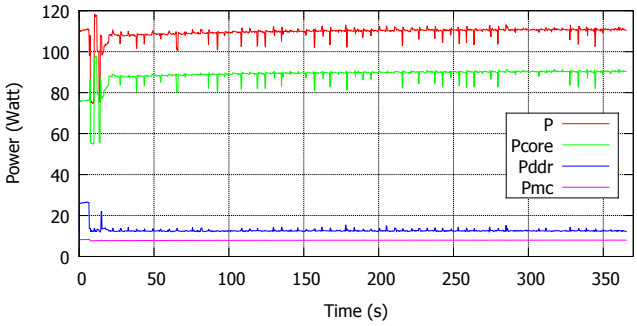
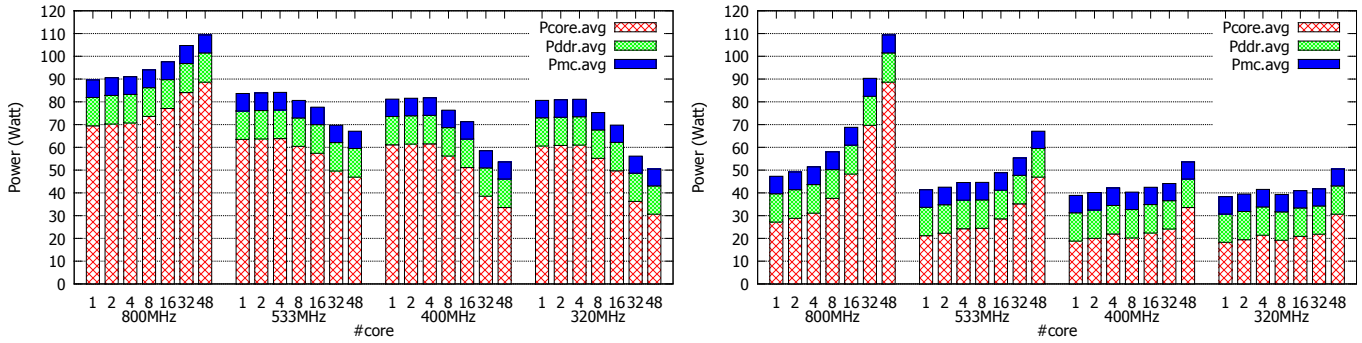



Fig. 2. The power characteristic of Intel SCC when Graph500 executes at 800MHz/1.1V on 48 cores

### A. Performance Characteristics

In order to evaluate the scalability performance of our Graph500 implementation, we present the comparison of runtime performance in Fig. 4, including the performance of two kernels of Graph500. For kernel 2, we present the average execution time of the 64 BFSes.

From the Fig. 4, we can find that, for fixed number of CPU cores, the execution time usually increases in lower power state. Likewise, for the same power state, the performance increases with the number of cores. Compared with 1 core, execution on 48 cores achieves average 1.69x, 4.10x and 3.99x maximal speedup for the whole benchmark, kernel 1 and kernel 2 respectively. However, sometimes, the whole benchmark (in Fig.4(a)) cannot achieve the best runtime performance on 48 cores. For example, at 800MHz/1.1V and 320MHz/0.8, the whole Graph500 on 48 cores is a little slower than 32 cores. This might be due to the performance overhead of the SPM or




(a) Measured power

(b) Estimated power excluding the standby dynamic power

Fig. 3. Power comparison of Graph500 at different power states on different number of cores

that the problem size is not large enough. This phenomena is also observed in kernel 1 and kernel 2. The whole benchmark achieves the best speedup (up to 1.72x) at 400MHz on 48 cores, while 4.65x for kernel 1 at 320MHz on 32 cores and 4.23x for kernel 2 at 400MHz on 32 cores.

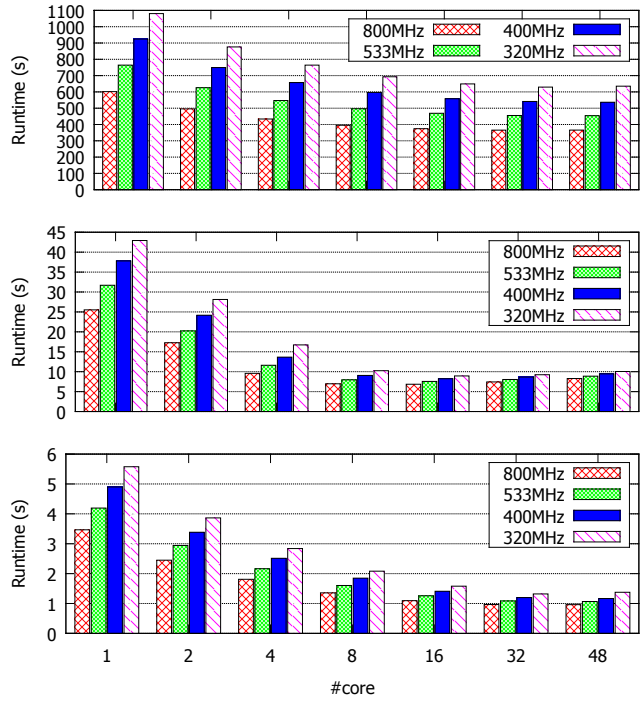



Fig. 4. Performance comparison of Graph500 at different power states on different number of cores. From up to down, (a) Execution time of the whole Graph500; (b) Execution time of kernel 1; (c) The average execution time of kernel 2

## B. Power Characteristics

Fig. 3 presents the power characteristics of Graph500 on different number of cores at different power states. Fig. 3(a) is the measured power. We present the average power during the execution of $P_{core}$, $P_{ddr}$ and $P_{mc}$. However, someone would say that the power of non-activated cores (or standby cores) should not counted in the comparison. Thus we estimate the per-core dynamic power of standby cores and exclude these power for comparison[1], as shown in Fig. 3(b).

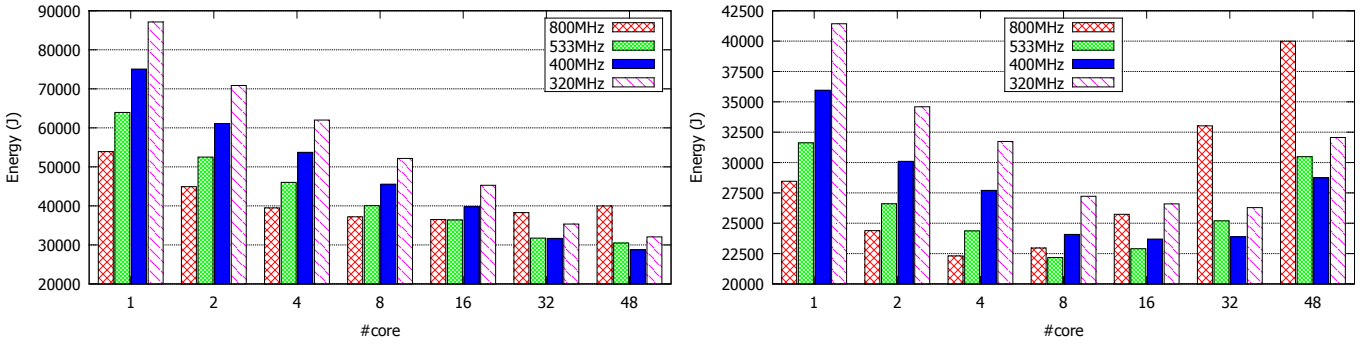
In both Fig. 3(a) and Fig. 3(b), the power at 800MHz is the most representative. The whole power $P$ almost linearly increases with the number of cores as higher power state consumes more power. For the same number of core, e.g. 48 cores, lower power state achieves smaller $P$ while some portion of the power (e.g. $P_{ddr}$ and $P_{mc}$) keep the same as they are not affected by the DVFS. The highest $P$ could reach up to 110 Watts at 800MHz on 48 cores.

## C. Energy and EDP Characteristics

Energy consumption and energy delay product (EDP) is two important power efficiency metrics [10]. Fig. 5 presents the comparison of energy, while Fig. 6 presents the comparison of EDP.

In Fig. 5(a), if we consider the standby power as the system power, we always get the less energy on more cores for the same power state (800MHz is an exception). This is because the standby power is even larger than these three power states. Thus excluding the standby dynamic power is more meaningful and reasonable. Fig. 5(b) presents the energy comparison after excluding the standby dynamic power. For fixed number of core, the best power state for the least energy is different. For example, 400MHz/0.8V is the best for 32 and 48 cores while 533MHz for 8 and 16 cores. For fixed power state, the best number of cores for the least energy is also different. Lower power state always has larger number of cores for the best result. 32 cores is the best for 320MHz, while four cores for 800MHz. Execution on 8 cores at 533MHz achieves the least energy, 46.48% saving compared with the largest energy on 1 core at 320MHz.

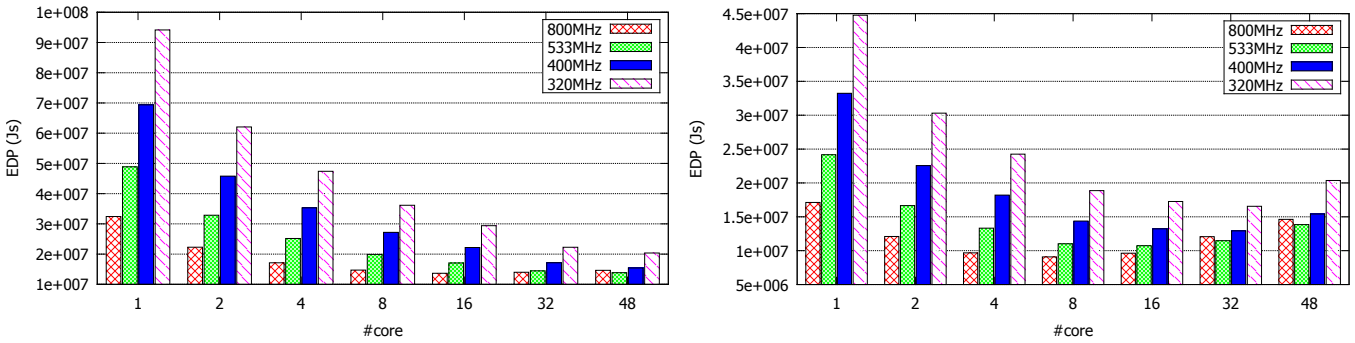
[1]As the standby cores are initialized to 533MHz/1.1, we firstly measure the $P_{core}$ at this power state (it's about 66.95 Watts). Then we set all the CPU cores to the lowest power state (100MHz/0.8V) which is safe and supported by Intel SCC [9]. The $P_{core}$ is about 23.75 Watts. Thus for each core, the reducible dynamic standby power is (66.95-23.75)/48 = 0.90 Watts.




(a) Energy with measured $P$

(b) Energy with estimated $P$ excluding the standby dynamic power

Fig. 5. Energy comparison of Graph500 at different power states on different number of cores




(a) EDP with measured $P$

(b) EDP with estimated $P$ excluding the standby dynamic power

Fig. 6. EDP comparison of Graph500 at different power states on different number of cores

For EDP, the situation is a little different because in this case the execution time is more weighted. As shown in Fig. 6(b), the best power state for the least EDP is 533MHz for 32 and 48 cores. While the best number of cores is still 32 cores for 320MHz but eight cores for 800MHz. Execution on 8 cores at 800MHz achieves the least EDP, 79.71% saving compared with the largest EDP on 1 core at 320MHz. Thus, we can say that the optimal power state or number of cores are decided by the optimization goals, e.g. the least energy or EDP.

## V. Conclusion

In this paper, we present a data-level paralleled implementation of Graph500 on Intel SCC. The parallelism follows a shared virtual programming model exploiting the shared physical memory (SPM) on SCC. We thoroughly evaluate the power and performance characteristics of Graph500 on various scale of cores at different power states. The experimental results show that the kernels of Graph500 could achieve up to 4.65 times performance speedup. Appropriate scale of cores and voltage/frequency setting could achieve up to 46.48% energy saving and up to 79.71% EDP reduction. We believe that the work provides valuable insight for designing an energy-efficient many-core system for data-intensive applications.

## Acknowledgment

This work is supported by Hong Kong RGC grant HKU 716712E, Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT, No. IRT1012) and Aid Program for Science and Technology Innovative Research Team in Higher Educational Institutions of Hunan Province (No. 11JJ7003). Special thanks go to Intel China Center of Parallel Computing (ICCPC) and Beijing Soft Tech Technologies Co., Ltd. for providing us with support services of the SCC platform in their Wuxi data center.

## References

[1] "Graph 500 benchmark." [Online]. Available: http://www.graph500.org

[2] F. Checconi and F. Petrini, "Massive data analytics: the graph 500 on IBM Blue Gene/Q," *IBM J. Res. Dev.*, vol. 57, no. 1, pp. 111–121, 2013.

[3] T. Gao, Y. Lu, and G. Suo, "Using MIC to accelerate a typical data-intensive application: The breadth-first search," in *Proc. the 27th Int. Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*, 2013, pp. 1117–1125.

[4] J. Jose, S. Potluri, K. Tomko, and D. Panda, *Designing Scalable Graph500 Benchmark with Hybrid MPI+OpenSHMEM Programming Models*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7905, ch. 9, pp. 109–124.

[5] "Top500 list - June 2014." [Online]. Available: http://www.top500.org/lists/2014/06/

[6] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, and K. Yelick, "A view of the parallel computing landscape," *Commun. ACM*, vol. 52, no. 10, pp. 56–67, 2009.

[7] "SCC external architecture specification (EAS) (revision 0.94)," Intel Labs, Tech. Rep., 2010.

[8] K. T. Lam, J. Shi, D. Hung, C.-L. Wang, Y. Yan, and W. Zhu, "Rhymes: A shared virtual memory system for non-coherent tiled many-core architectures," in *Proc. 20th IEEE Int. Conf. Parallel and Distributed Syst. (ICPADS'14)*, 2014, (in press).

[9] Z. Lai, K. T. Lam, C.-L. Wang, J. Su, Y. Yan, and W. Zhu, "Latency-aware dynamic voltage and frequency scaling on many-core architectures for data-intensive applications," in *Proc. Int. Conf. Cloud Computing and Big Data (CloudCom-Asia'13)*, 2013, pp. 78–83.

[10] V. W. Freeh, D. K. Lowenthal, F. Pan, N. Kappiah, R. Springer, B. L. Rountree, and M. E. Femal, "Analyzing the energy-time trade-off in high-performance computing applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 6, pp. 835–848, 2007.