The Energy Consumption Analysis for the Multispectral Infrared Satellite Images Processing Algorithm

Ekaterina Tyutlyaeva ✉, Sergey Konyukhov, Igor Odintsov, and Alexander Moskovsky

ZAO RSC Technologies,
Kutuzovskiy av., 36, building 23, 121170 Moscow, Russia
{xgl,s.konyuhov,moskov,igor_odintsov}@rsc-tech.ru

Abstract. This paper includes the energy consumption analysis of the testing mini-application that implements night time infrared remote sensing algorithm `Nightfire`. On this stage of our project computational nodes with Intel Xeon E5 and Intel Xeon Phi processors were tested.
The correlation analysis between the number of used MPI ranks - OMP threads and total energy consumptions was performed for each of tested computational nodes. The optimized benchmarking parameters were used to compare energy efficiency of tested nodes.
Moreover, the analysis of mini-application statements blocks was carried out for the following computation phases: I/O with HDF5 and ENVI data; the data processing using Nelder-Mead method. The impact of each computation phase to the total energy consumptions was determined so it gives new insights to possible ways of further optimization.
Based on obtained results, the effectiveness of tested computational architectures for multispectral satellite images processing was evaluated.

Keywords: Energy Consumption ·Energy Efficiency ·Satellite Image Processing ·Power-Aware Execution ·Hybrid Parallel Programming

1 Introduction

Nowadays power-aware program executions are one of the most immediate and pressing challenges facing software developers. The software energy-efficient design and execution tuning are especially relevant to the reusable applications that are executed regularly because, among other things, energy consumption depends on the algorithmic structure and computation phases. Thereby, the reasonable strategy of hardware usage by program application can result in substantial reduction of supercomputer operational costs. In that context, the power consumption during the application runtime should be explored, and the most power efficient execution configuration should be suggested before the productive use of software.

In this research we conducted the energy-efficiency analyses of `Nightfire` algorithm for multispectral infrared satellite images (MISI) [1]. The multispectral

data for this algorithm are collected globally, each night, by the Visible Infrared Imaging Radiometer Suite (VIIRS) [2] operated by Suomi National Polar Partnership (NPP) [3]. Thus, software implementing `Nightfire` algorithm is planned to be used on a daily basis, and so it should be tuned with power concerns.

In our tests we used own mini-application implementing `Nightfire` algorithm by so-called hybrid programming model that combines shared-memory (OpenMP) and distributed-memory (MPI) programming models. The hybrid programming model allows to use many- and multi-core processors in more efficient way and provides opportunities for flexible application tuning against target architectures. Besides, the hybrid programming model can utilize all available computing cores and RAM resources more efficiently than pure MPI model what may cause a program to run faster [4].

On the other hand, due to the flexible nature of hybrid programming model it is possible to optimize program runtime parameters - first of all, the numbers of MPI processes and OMP threads - regarding not only to the execution time, but also to the power consumption.

Finally, drawing up energy profiles along the critical path of program execution allows to identify those blocks of the studied algorithm that incur the most energy consumption, which helps to determine further directions of code optimization.

2 Related work

Power-aware execution of parallel programs is now a primary concern in large-scale HPC environments. Dong Li at al. [5] presented and evaluated solutions for power-efficient execution of programs written in the hybrid program model targeting large-scale distributed systems with multicore nodes. The authors used a new power-aware performance prediction model of hybrid MPI/OpenMP applications to derive a novel algorithm for power-efficient execution of realistic applications.

Power-aware policies were the focus of many researchers. Typically, the studies proposes to expand the time-saving strategies with power-saving features. As an example, Wenlei Bao et al. [6] proposed a novel power-aware WCET (Worst Case Execution Time) analysis technique to improve system energy efficiency and simultaneously validate real-time tasks. The Power-Aware Linear Programming based Affinity Scheduling Policy was described in [7].

In this paper we studied energy efficiency of MISI processing on the various Intel Xeon E5 and Intel Xeon Phi architectures. We carried out cross-architectural research that includes computation phase analysis and CPU/DRAM power consumption distribution. Our methodology agrees with common profiling approach and can be used for wide range of software to give recommendations for power-aware executions.

3 Algorithm

For our energy consumption analysis we used own mini-application implementing `Nightfire` algorithm for the multispectral infrared satellite images processing. This algorithm detects and characterizes sub-pixel hot sources using multispectral data collected for different infrared spectral ranges. `Nightfire` algorithm relies on the approximation of direct composition of theoretical Planck curves describing radiation of subpixel heat source(s) and sea/land surface temperature background. The Nelder-Mead simplex algorithm is used for the unconstrained nonlinear fit of a weighted mixture of the Planck curves to observed multispectral radiances.

The method can be used to process the data from the last generation of infrared sensors at the environmental satellites: VIIRS (solar synchronous Suomi NPP and JPSS-1, NOAA/NASA), ABI (geostationary GOES-R, NOAA) [8], Landsat 8 (low orbiting, USGS) [9], and AHI (geostationary Himawari 8-9, JMA) [10].

The basic flowchart illustrating the structure of `Nightfire` algorithm is shown on the Fig. 1.

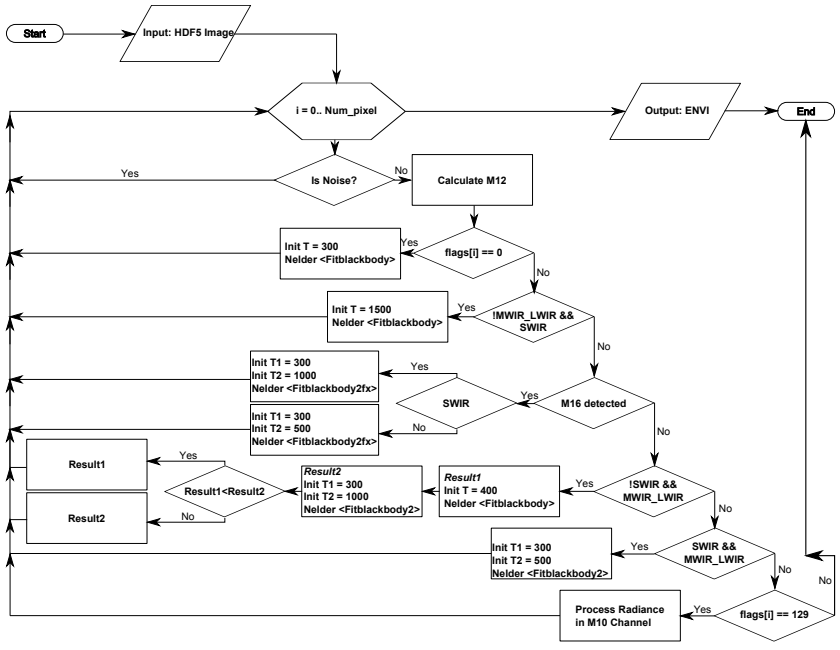


Fig. 1. `Nightfire` Algorithm Flowchart

`Nightfire` algorithm has the high scalability potential because it processes images by applying Nelder-Mead optimization method [11] to individual image

pixels independently from each other. Moreover, the most computational intensive part of Nelder-Mead optimization method, namely, the evaluation of target function values, is also well suited for parallelization. Nevertheless, huge computational demands of this task stem from the necessity to process large number of images with large number pixels during short time period.

We used the heterogeneous MPI+OMP parallelization scheme to effectively utilize the multicore architectures. The manual and the compiler-supported code vectorization for Intel architectures also were applied before the analysis stage.

4 Hardware

In the table 1 codenames and specifications of studied testbeds are listed.

Table 1. Testbeds Specifications

Codename	CPU	# Cores	Memory	GB per Core
Haswell	Intel Xeon E5-2697 v3	2x 14	8x DRAM Samsung 16GB DDR4/2133MHz	4.57
Broadwell	Intel Xeon E5-2697A v4	2x 16	8x DRAM Samsung 16GB DDR4/2133MHz	4
KNL	Intel Xeon Phi 7250	68	MCDRAM Intel 16GB + 6x DRAM Micron 32GB DDR4/2133MHz	2.8

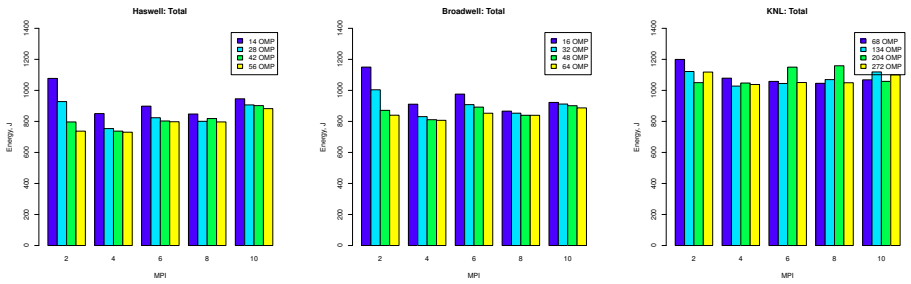
5 Measurements

The energy consumption of the tested mini-application was studied using the Intel Running Average Power Limit (RAPL) counters [13]. RAPL provides a way to measure the power consumption on processor packages and DRAM. According to the recent studies this software power model matches the actual power measurements [14]. Further all results are given as averaged results of the multiple executions.

PAPI performance application programming interface [15] was used to gather RAPL data statistics. The distribution of total energy consumption for different MPI/OMP numbers are shown on the Fig. 2.

We used the following execution parameters: 2/4/6/8/10 MPI processes and 1xN/2xN/3xN/4xN OMP threads (in total), where N is the number of cores per

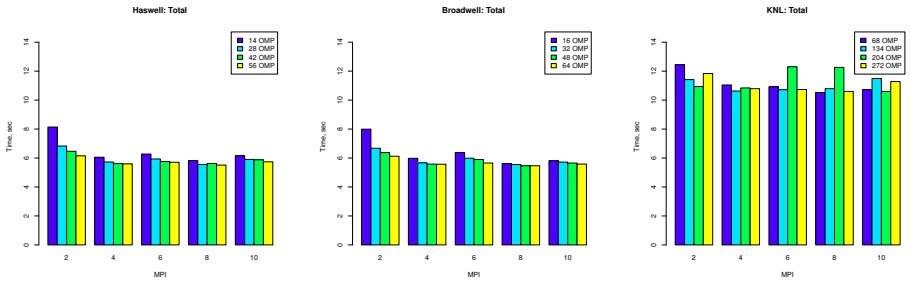
one CPU. Consequently, due to the usage of Intel Hyper-Threading Technology for Intel processors [12] the maximal number of OMP threads per CPU core was 56 for Haswell (2 (CPUs per node) × 14 (Cores per CPU) × 2 (Hyper-threading)); 64 for Broadwell (1 (CPU per node) × 14 (Cores per CPU) × 2 (Hyperthreading)) architectures, and 272 (2 (CPUs per node× 68 (Cores per CPU) × 4 (Hyperthreading) for KNL architecture.



a) b) c)

Fig. 2. Estimated Energy Consumption for DRAM and CPU Packages on a) Haswell, b) Broadwell, c) KNL architectures

The bar graph on the Fig. 2 expresses the total execution time in seconds for the tested mini-application.



a) b) c)

Fig. 3. Total Execution Time on a) Haswell, b) Broadwell, c) KNL architectures

The minimal energy consumption results for the tested mini-application are presented in table 2.

Obtained results show that for all testbeds the most optimal processes/threads configuration was 4 MPI/2 × Num_{cores} OMP threads per node, where Num_{cores} is the number of physical cores in a computational node.

Table 2. Minimal energy consumption for the tested architectures, J

Application	Haswell	Broadwell	KNL
DRAM	38.9043 (4 MPI 56 OMP)	184.749 (8 MPI 64 OMP)	27.071333 (8 MPI 272 OMP)
Processors	691.584 (4 MPI 56 OMP)	618.51 (4 MPI 64 OMP)	995.386667 (4 MPI 136 OMP)
Total	730.488 (4 MPI 56 OMP)	807.076 (4 MPI 64 OMP)	1027.73 (4 MPI 136 OMP)

While there is some correlation between the execution time and the consumed energy, it has complex noninear character. The values of energy (ETS) and time (TTS) to solve test problem for all testbeds are presented in the tables 3, 4, 5.

Table 3. Energy (ETS, J) and time (TTS, sec) obtained for the test on Haswell testbed with different MPI and OMP configurations

MPI	Type	14 OMP	28 OMP	42 OMP	56 OMP
2	TTS	8.139	6.829	6.466	6.149
	ETS	1077.12	927.882	796.54	737.235
4	TTS	6.055	5.726	5.619	5.596
	ETS	850.526	753.846	737.405	730.488
6	TTS	6.276	5.935	5.762	5.706
	ETS	898.315	823.753	802.762	797.944
8	TTS	5.814	5.558	5.624	5.514
	ETS	847.626	800.677	818.487	796.684
10	TTS	6.151	5.904	5.879	5.738
	ETS	945.801	906.363	901.844	882.429

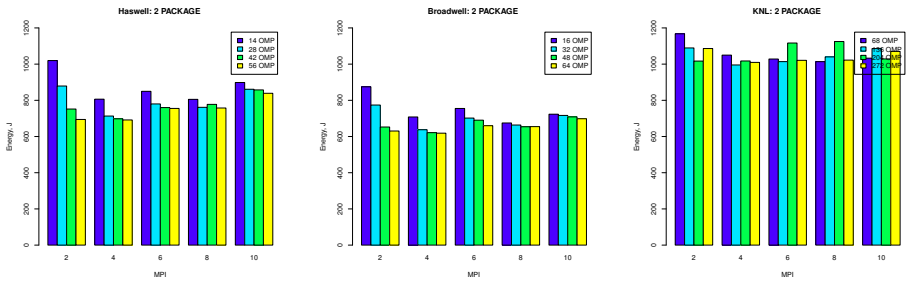
For example, the most power-aware MPI/OMP execution configuration for Haswell processor took 5.596 sec and 730 J. The most time-efficient (fast) result was 5.514 sec, but the energy consumption for this configuration was 796 J, that is 66 J more then the energy efficient one.

For Broadwell processor, the most energy efficient result was 5.571 sec for runtime and 807 J for energy consumption.

It is worth noting that, while Haswell architecture demonstrated the best total energy efficient result, the best results for CPU energy consumption were yielded on Broadwell testbed (Fig. 2).

Table 4. Energy (ETS, J) and time (TTS, sec) obtained for the test on Broadwell testbed with different MPI and OMP configurations

MPI	Type	16 OMP	32 OMP	48 OMP	64 OMP
2	TTS	7.993	6.674	6.375	6.1236
	ETS	1150.2	1003.82	871.343	840.002
4	TTS	5.979	5.664	5.581	5.571
	ETS	910.977	830.855	811.053	807.076
6	TTS	6.373	5.987	5.897	5.654
	ETS	975.988	908.189	892.213	852.781
8	TTS	5.611	5.549	5.462	5.465
	ETS	866.233	853.055	839.569	839.433
10	TTS	5.805	5.719	5.653	5.582
	ETS	922.358	912.005	901.284	887.115



a) b) c)

Fig. 4. CPU Energy Consumption for a) Haswell, b) Broadwell, c) KNL architectures

However, the energy-efficiency of Broadwell processors was neglected by DRAM consumption (Fig. 2). Broadwell architecture has demonstrated the highest rates of DRAM energy consumption, while KNL DRAM energy consumption is the lowest one.

The difference in DRAM energy consumption between Haswell and Broadwell architectures was even more remarkable because DRAM models for these nodes were absolutely identical.

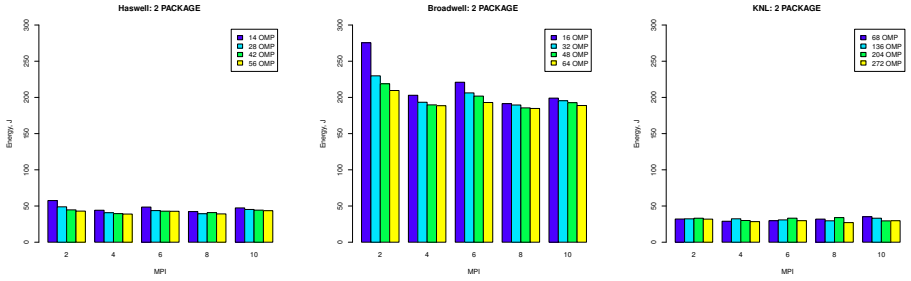
6 Computation Stages

6.1 Input/Output

For this work, the mini-example of satellite image was prepared in HDF5 data format [16]. Input data contained 9830400 pixels, file size was 236 Mb. The HDF5 I/O library supports parallel I/O, so we have used parallel data reading operations.

Table 5. Energy (ETS, J) and time (TTS, sec) obtained for the test on KNL testbed with different MPI and OMP configurations

MPI	Type	68 OMP	136 OMP	204 OMP	272 OMP
2	TTS	12.441	11.417	10.935	11.838
	ETS	1199.53	1049.95	1118.04	
		1121.73			
4	TTS	11.045	10.626	10.841	10.788
	ETS	1078.75	1027.73	1047.4	1037.99
6	TTS	10.921	10.716	12.300	10.732
	ETS	1057.9	1044.61	1149.79	1050.69
8	TTS	10.518	10.794	12.257	10.598
	ETS	1045.54	1069.55	1158.77	1049.24
10	TTS	10.724	11.491	10.600	11.281
	ETS	1068.12	1118.9	1058.22	1100.22



a) b) c)

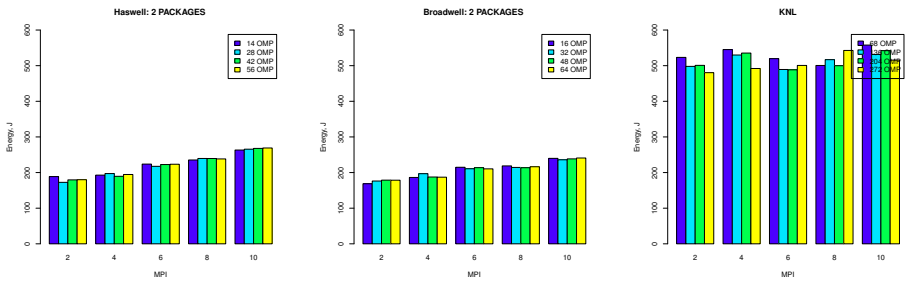
Fig. 5. DRAM Energy Consumption for a) Haswell, b) Broadwell, c) KNL architectures

However, the processors energy consumption increases steadily with the number of used MPI processes for all architectures (see Fig. 6):

- Haswell testbed – 179,5 J in average for 2 MPI executions, 265,75 J for 10 MPI executions;
- Broadwell testbed – 175 J in average for 2 MPI executions, 238 J for 10 MPI executions;
- KNL testbed – 500,25 J in average for 2 MPI executions, 536 J for 10 MPI executions.

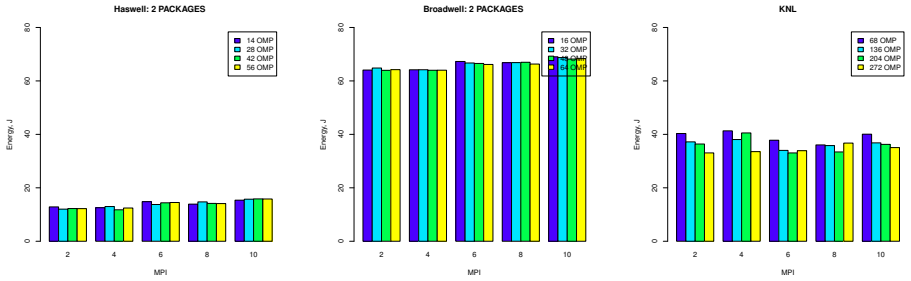
DRAM energy consumption also increased with the growing number of MPI ranks used (see Fig. 7).

The output data were prepared in ENVI format and took about 300 Mb of disk space. While ENVI output weren't parallelized, KNL energy consumption was comparable to Haswell and Broadwell energy consumption for this phase



a) b) c)

Fig. 6. CPU Energy Consumption during the Input Stage on a) Haswell, b) Broadwell, c) KNL architectures



a) b) c)

Fig. 7. DRAM Energy Consumption during the Input Stage on a) Haswell, b) Broadwell, c) KNL architectures

of algorithm and took about 250 J (vs 500 J for HDF5 input) (see Fig. 8). Consequently, for Intel Xeon Phi architecture special attention should be drawn to HDF5 input phase.

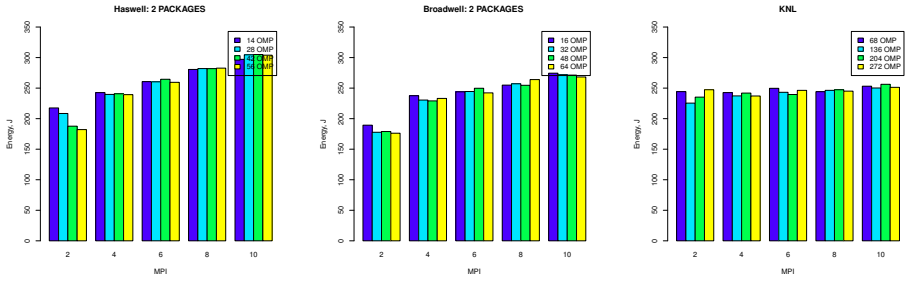
6.2 Image Processing

On the contrary to the input phase, for the image processing stage the test run with 2 MPI ranks was the least successful case in energy consumption terms for processors (see Fig. 9) as well as DRAM (Fig. 10).

It is also worth noting that for this stage of algorithm the energy consumption difference between test runs with different MPI ranks was significant, so the less efficient configurations should be avoided.

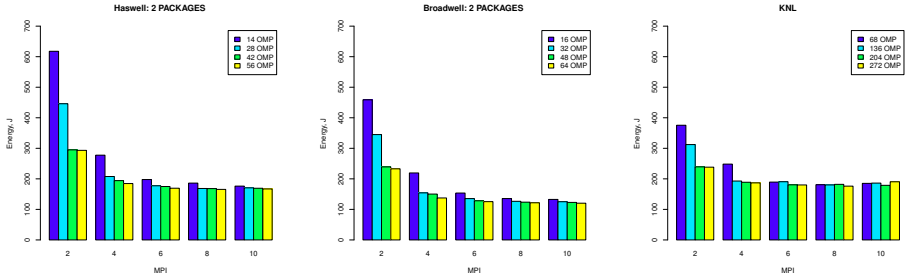
The most power efficient MPI/OMP cases on the most computation demanding stage (2 MPI 14 OMP for Haswell and 2 MPI 16 OMP for Broadwell respectively) didn't utilize all computation resources [1], that leads to the execution time and the total energy consumption increasing.

[1] If it be taken into account the hyperthreading technology



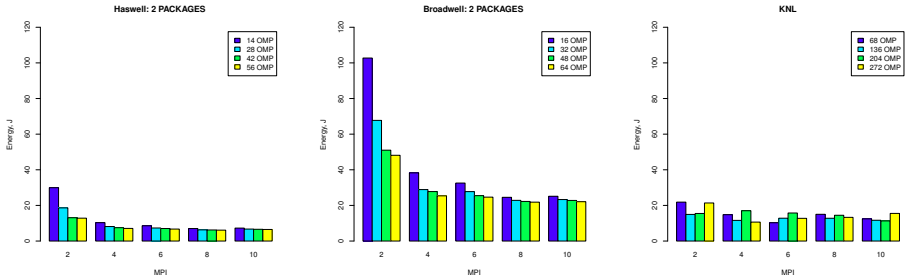
a) b) c)

Fig. 8. CPU Energy Consumption during the Output Stage on a) Haswell, b) Broadwell, c) KNL architectures



a) b) c)

Fig. 9. CPU Energy Consumption during the Image Processing Stage on a) Haswell, b) Broadwell, c) KNL architectures



a) b) c)

Fig. 10. DRAM Energy Consumption during the Image Processing Stage on a) Haswell, b) Broadwell, c) KNL architectures

The energy effective MPI/OMP combinations can be proposed as follows:

- The best configurations for Haswell testbed include 8/10 MPI, 28/42/56 OMP cases (The results varies in ranges 165-170 J per CPUs, 6.1-6.7 J per DRAM).
- The best configurations for Broadwell testbed include 8/10 MPI, 32/48/64 OMP cases (The results varies in ranges 120-126 J per CPUs, 22-23 J per DRAM).
- The best possible options for KNL cover 6/8/10 MPI, 68/136/204/272 OMP cases (The results varies in ranges 180-190 J per CPU, 11-15 J per DRAM).

It is worth mention that for the input/output stages the energy consumption was higher than for the image processing stage. This difference is particularly notable for KNL testbed, so the major optimization efforts should be aimed to the input/output phases before the processing of big amounts of data. We plan in the future work to study the energy-efficiency of possible models of the usage of HBM[2] on KNL testbeds.

7 Conclusions

The results of our test runs have showed that the isolated CPU energy consumption didn't reflect a full picture. For example, for Broadwell testbed processors have showed the most power-aware execution results, but DRAM energy consumption was the substantial and the largest among the considered testbeds share of the total energy consumption .

Our experimental studies have demonstrated the efficiency of simple energy measurements to reveal energy consumption flaws. As given test results illustrate, the execution time could be slightly different for different runtime configurations, but the energy consumption for them might change considerably.

So it seems to be possible to tailor suitable TTS/ETS configurations for each satellite processing algorithm in order to reduce the total energy consumption. For `Nightfire` algorithm of MISI processing we can propose 4 MPI/$2\times Num_{cores}$ configurations as the most power-aware configurations.

In the future, we plan to continue these experiments with larger amounts of real satellite data (up to 20-30 TB) and proof the scalability of our assumptions.

Currently, the VIIRS boat detector algorithm is developed for subsequent study. It is is an innovative method to robustly identify fishing boats at night using visible and infrared images from the SNPP satellite. We are planning to study the inter-node communications and larger amount of data using this algorithm in the nearest future.

Acknowledgments. This research was supported by the Russian Federal Science and Technology Program grant 14.607.21.0165 Efficient co-design of massively parallel computer for multispectral night-time remote sensing.

[2] On-package high-bandwidth memory based on the multi-channel dynamic random access memory, MCDRAM

References

1. Elvidge, C.D.; Zhizhin, M.; Hsu, F.-C.; Baugh, K.E. VIIRS Nightfire: Satellite Pyrometry at Night. Remote Sens. 2013, 5, 4423-4449.
2. Cao, C., X. Shao, X. Xiong, S. Blonski, Q. Liu, S. Uprety, X. Shao, Y. Bai, F. Weng, Suomi NPP VIIRS sensor data record verification, validation, and long-term performance monitoring, Journal of Geophysical Research: Atmospheres, DOI: 10.1002/2013JD020418, 2013.
3. Suomi NPP (National Polar-orbiting Partnership), Home Page, Accessed: 13.04.2017 `http://rammb.cira.colostate.edu/projects/npp/`
4. Nicholas J Wright, Karl Fuerlinger, Hongzhang Shan, Tony Drummond, Andrew Canning, and John Shalf: Best Practices for Hybrid OpenMP/MPI Programming on Hopper. The Cray Center of Excellence: Performance Optimization for the Multicore Era //NERSC/LBNL, Princeton Plasma Physics Lab, Oct 2010; slides `https://www.nersc.gov/assets/NUG-Meetings/NUG-Oct2010-Wright.pdf`
5. D. Li, B. R. de Supinski, M. Schulz, K. Cameron and D. S. Nikolopoulos, "Hybrid MPI/OpenMP power-aware computing," 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS), Atlanta, GA, 2010, pp. 1-12. doi: 10.1109/IPDPS.2010.5470463
6. W. Bao, S. Tavarageri, F. Ozguner and P. Sadayappan, "PWCET: Power-Aware Worst Case Execution Time Analysis," 2014 43rd International Conference on Parallel Processing Workshops, Minneapolis, MN, 2014, pp. 439-447. doi: 10.1109/ICPPW.2014.64
7. Hadil Al-Daoud, , Issam Al-Azzoni, , Douglas G. Down, Power-aware linear programming based scheduling for heterogeneous computer clusters // Future Generation Computer Systems, Volume 28, Issue 5, May 2012, Pages 745-754
8. NOAA GOES-R Web Site // A collaborative NOAA & NASA program, Accessed: 13.04.2017 `http://www.goes-r.gov/`
9. LANDSAT 8 (L8) DATA USERS HANDBOOK, LSDS-1574, Sioux Falls, South Dakota, March 29, 2016; Accessed: 13.04.2017 `https://landsat.usgs.gov/sites/default/files/documents/Landsat8DataUsersHandbook.pdf`
10. Himawari User's Guide, Home Pagev// Japan Meteorological Agency (JMA) Accessed: 13.04.2017 `http://www.jma-net.go.jp/msc/en/support/index.html`
11. J.C. Lagarias, J.A. Reeds, M.H. Wright, and P.E. Wright, Convergence properties of the Nelder-Mead simplex method in low directions, SIAM J. Optim., 9 (1998), pp 112-147
12. Intel Hyper-Threading Technology, `http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html`, (accessed: 10.04.2017)
13. Intel 64 and IA-32 Architectures Software Developers Manual, Volume 3B: System Programming Guide, Part 2, `http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.pdf`
14. E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann and D. Rajwan, "Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge," in IEEE Micro, vol. 32, no. 2, pp. 20-27, March-April 2012. doi: 10.1109/MM.2012.12
15. Jagode, H., YarKhan, A., Danalis, A., Dongarra, J. "Power Management and Event Verification in PAPI," 9th Parallel Tools Workshop, Dresden, Germany, September 2-3, 2015.

16. HDF5 Tutorial, Parallel Topics. https://support.hdfgroup.org/HDF5/Tutor/parallel.html