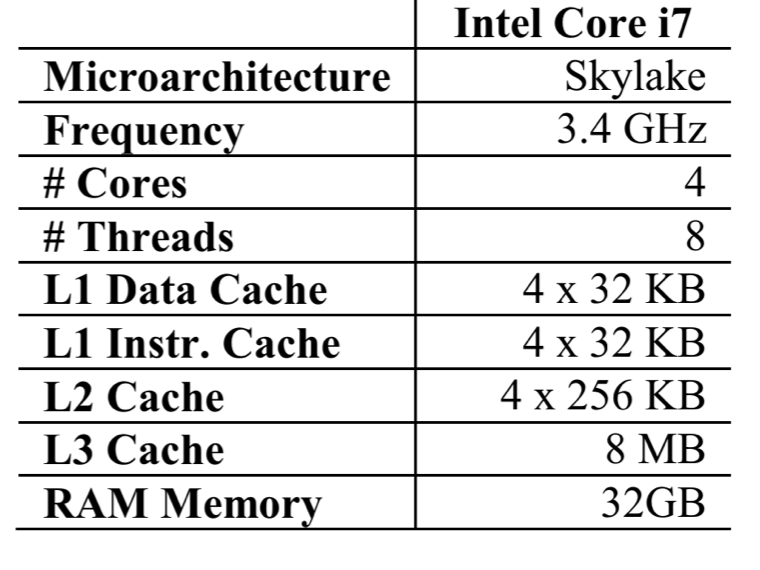
**Data Extraction SLR Papers**

**IEEE:**

**Paper1: How Programming Languages and Paradigms Affect Performance and Energy in Multithreaded Applications**

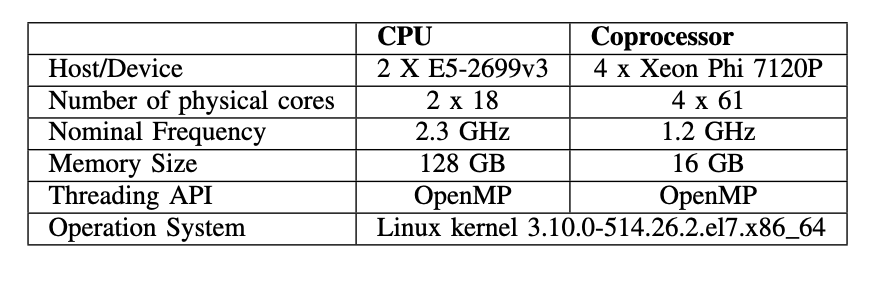
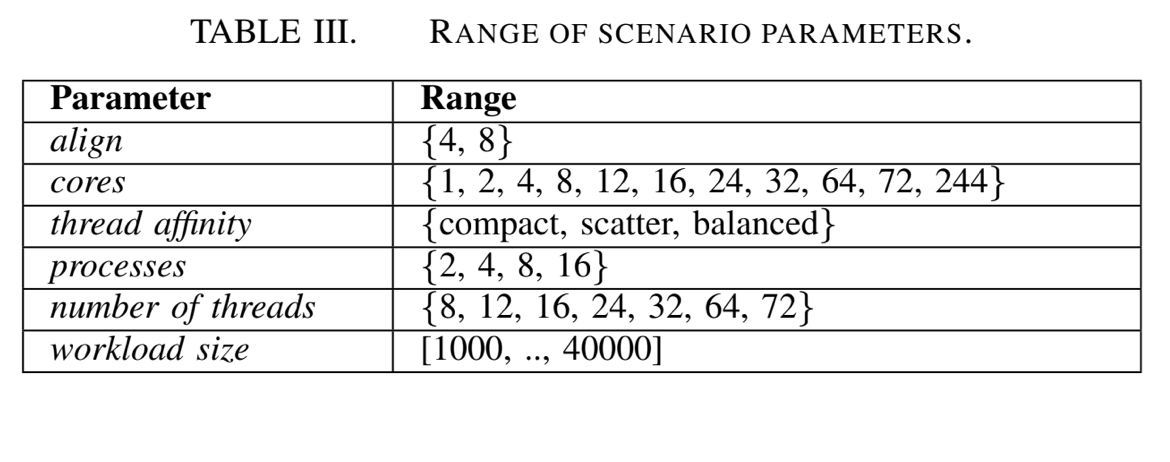
* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* NAS Parallel Benchmarks
  + Fourier Transformation
  + Integer Sort
  + Conjugate Gradient
  + Low-Upper
* For Java: The synchronization between the threads is guaranteed by the use of wait/notify, and mutual exclusion (synchronized blocks and methods).
* The C++ applications were implemented using the C++11 Threads library, which is based on C's library POSIX Threads, following object-oriented guidelines.To synchronize, mutual exclusion and condition variables were used; and to ensure data integrity on parallel regions, atomic variables were used.
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware:

* OS: Ubuntu 16.04 kernel 4.4.0-21.
* The C and C++ compiler used was the GNU GCC-4.9 with the optimization flag -O3.
* The OpenMP version used in the C applications was the 4.0, since it is fully supported by the GCC version in use. Java was executed using the Oracle 1.8.0\_10.To measure the influence of the JIT in the Java versions (which is enabled by default), we disabled it by using the flag “-Xint”.
* Energy consumption obtained: RAPL, using the following: package, which provides the energy of the whole CPU package (core, cache, bus, etc.); and the memory controller DRAM
* Threads split: 1,2,3,4,8
* What is/are the variables authors are observing apart of energy consumption, if any?
* Number of threads over the time for different applications and 4 languages: Java, Java - JIT, C, C++
* Energy over Number of threads
* EDP = Energy \* Delay
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* average of ten executions, with a standard deviation lower than 1% for each benchmark. (mean)
* Does there exist a replication package for the reported experiments?
* No replication package available for this paper

Paper7: Towards New Metrics for Appraising Performance and Energy Efficiency of Parallel Scientific Programs

* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* PARSEC
  + blackscholes
  + bodytrack
  + canneal
  + dedup
  + facesim
  + ferret
  + fluid
  + freqmine
  + streamcluster
  + swaptions
  + vips
  + x264
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
  + Hardware: Intel Core i7-4770 with the Haswell architecture and an Intel Core i7-6700 with the Skylake architecture.The usage of Intel Turbo Boost was disabled on both systems. The processors have four physical cores with hyper-threading, leading to eight logical cores. The memory hierarchy includes an 8 MB shared L3 cache as well as a 256 KB L2 cache and a 32 KB L1 cache per core. The main memory size is 16 GB. The specified thermal design power (TDP) is 84 W on the Haswell system and 65 W on the Skylake system.
  + Energy tool: RAPL
  + To set the frequencies of the cores to a fixed value, the cpufreq\_set tool has been used.
  + Threads: 1,2,4,8
  + Freq: 0.8-3.4
* What is/are the variables authors are observing apart of energy consumption, if any?
* Power - frequency
* Threads - frequency
* Speedup - threads
* energy speedup and runtime speedup
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* Not mentioned how much experiments
* Does there exist a replication package for the reported experiments?
* Not found

Paper10: Performance and Energy Consumption Analysis of Coprocessors using Different Programming Models ( Not sure if HPC)

* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* Linpack and HPL 2.1 benchmarks
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware: 
* Energy tool: Intel Open Source RAPL and Power Cap Framework were used for the monitoring and collection of the average power consumption for both CPU and RAM. The available power counters can be accessed at */sys/class/powercap/intel-rapl/*.
* Parameters: 
* What is/are the variables authors are observing apart of energy consumption, if any?
* Performance (GFfops) - Workload size
* Threads - Energy
* Execution time - workload size
* Performance per watt - threads
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* not found
* Does there exist a replication package for the reported experiments?
* not found

Paper19: A Comprehensive Study on the Energy Efficiency of Java’s Thread-Safe Collections

* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* Java’s Thread-Safe Collections
  + Lists (java.util.List)
    - Iterator
    - ArrayList
    - Vector
    - Collections.synchronizedList()
    - CopyOnWriteArrayList
  + Maps (java.util.Map)
    - LinkedHashMap
    - Hashtable
    - Collections.synchronizedMap()
    - ConcurrentSkipListMap
    - ConcurrentHashMap
    - ConcurrentHashMapV8
  + Sets (java.util.Set)
    - LinkedHashSet
    - Collections.synchronizedSet()
    - ConcurrentSkipListSet
    - ConcurrentHashSet
    - CopyOnWriteArraySet
    - ConcurrentHashSetV8
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware:

System#1 :A2×16-core AMD Opteron 6378 processor

(Piledriver microarchitecture), 2.4GHz, with 64GB of DDR3

1600 memory. It has three cache levels (L1, L2, L3): L1 with

32KB per core, L2 with 256KB per core, and L3 20480 (Smart

cache). It is running Debian 3.2.46-1 x86-64 Linux (kernel

3.2.0-4-amd64), and Oracle HotSpot 64-Bit Server VM (build

21) JDK version 1.7.0 11.

System#2 :A2×8-core (32-cores when hyper-threading is

enabled) Intel(R) Xeon(R) E5-2670 processor, 2.60GHz, with

64GB of DDR3 1600 memory. It has three cache levels (L1,

L2, L3): L1 with 48KB per core, L2 with 1024KB per core,

and L3 20480 (Smart cache). It is running Debian 6 (kernel

3.0.0-1-amd64) and Oracle HotSpot 64-Bit Server VM (build

14), JDK version 1.7.0 71.

* Energy measured: jRAPL (only for System2) (system1 is power meter)
* RQ1. Do different implementations of the same collection have different impacts on energy consumption?
* RQ2. Do different operations in the same implementation of a collection consume energy differently?
* RQ3. Do collections scale, from an energy consumption perspective, with an increasing number of concurrent threads?
* RQ4. Do different collection configurations and usages have different impacts on energy consumption?
* For RQ1 and RQ2: number threads is 32
  + For the insertion operation, we start with an empty

collection, and each thread inserts 100,000 elements. At

the end the total number of elements inside the collection

is 3,200,000. To avoid duplicate elements, each insertion

operation adds a String with value thread-id + “-” +

current-index.

* For the traversal operation, each thread traverses the

entire collection generated by the insertion operation,

i.e., over 3,200,000 elements. On Sets and Maps, we

first get the list of keys inserted, and then we iterate

over these keys in order to get their values. On Lists,

it is performed using a top-level loop over the collection,

accessing each element by its index using the get(int

i) method.

* For the removal operation, we start with the collection

with 3,200,000 elements, and remove the elements one

by one, until the collection becomes empty. For Maps

and Sets, the removals are based on keys. On Lists,

the removal operation is based on indexes, and occurs

in-place — that is, we do not traverse the collection to

look up for a particular element before removal.

* RQ3: Map implementations only
* RQ3: Threads: (1, 2, 4,8, 16, 32, 64, 128, and 256 concurrent threads) and study how such variations impact energy consumption. An increment

in the number of threads also increments the total number

of elements inside the collection. Since each thread inserts

100,000 elements, when performing with one thread, the total

number of elements is also 100,000.

* What is/are the variables authors are observing apart of energy consumption, if any?
* Only energy such:
  + Energy - Map, Traversal, Removal
  + Energy - Number threads
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* We run each benchmark 10 times within the same JVM; this is implemented by a top-level 10-iteration loop over each benchmark. The reported data is the average of the last 3 runs.
* Does there exist a replication package for the reported experiments?
* Unfortunately no

Paper54: The Impact of Turbo Frequency on the Energy, Performance, and Aging of Parallel Applications

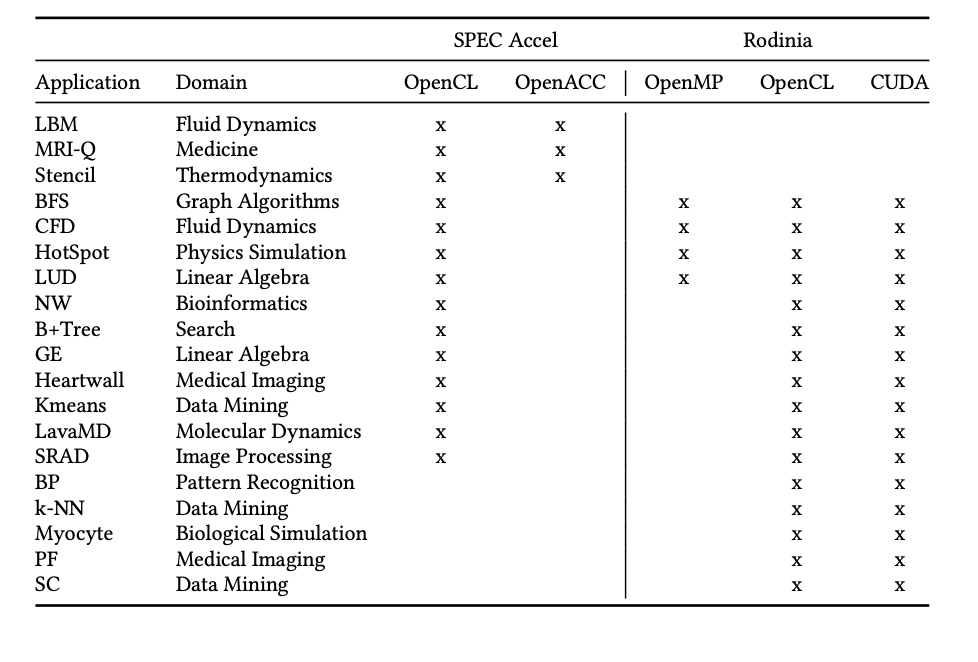
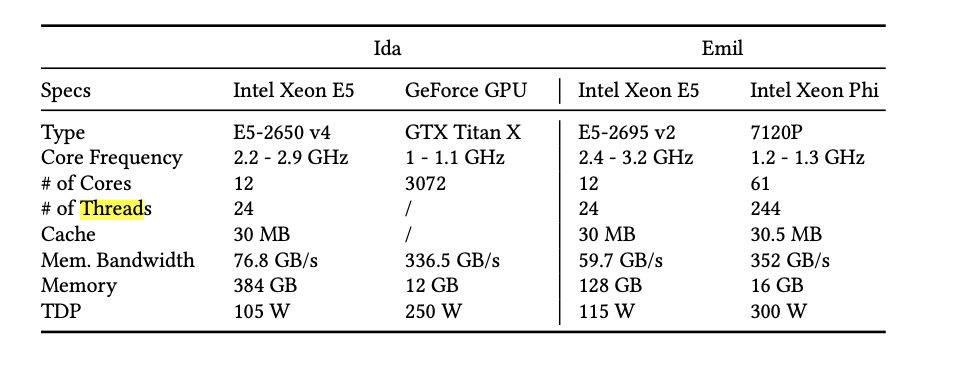
* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* The benchmarks used were: Seven kernels from the NAS Parallel benchmark [21]: block tri-diagonal solver (BT), conjugate gradient (CG), discrete 3D fast Fourier Transform (FT), lower-upper gauss- seidel solver (LU), multi-grid on a sequence of meshes (MG), scalar penta-diagonal solver (SP), and unstructured adaptive mesh (UA). As the original version of NAS is written in FORTRAN, we use the OpenMP-C version developed by the authors in [22].
* Two applications from the Rodinia Benchmark Suite [23]: *hotspot* (HS) and *streamcluster* (SC).
* Seven applications from different domains: *fast Fourier transform* (FFT) [24]; *the high performance conjugate gradient* (HPCG) benchmark [25]; *Jacobi Iteration Method* (JA) [26]; *Liver- more unstructured Lagrangian explicit shock hydrodynamics* (LULESH2) [27]; *n-body* (NB) [26]; *Poisson* (PO) [26]; and *STREAM* (ST)
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware: The experiments were performed on an AMD Ryzen 7 1700 (Zen) with SMT (16 threads total) running Ubuntu OS with Kernel v. 4.15. The normal operating frequency ranges from 1.5 GHz to 3.0 GHz when Turbo Core and Game Mode are turned off.
* Energy & Time: The execution time of each application was obtained through the *omp get wtime* function, while the Application Power Management library was used to get the energy consumption.
* Number of threads: 1,2,4,6,8,10,12,14,16
* What is/are the variables authors are observing apart of energy consumption, if any?
* Relative performance - number of threads
* Relative Energy - number of threads
* Relative EDP - number of threads
* Relative temp - number of threads
* Relative Aging - number of threads
* Measures above are on CPU and Memory
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* average of ten executions with σ à 0.5%.
* Does there exist a replication package for the reported experiments?
* Not found

Paper55: Automatic Energy-Efficiency Monitoring of OpenMP Workloads

* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* Rodinia benchmark
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware: It is tested on both an Intel Xeon dual-socket multicore system and an Odroid XU3 platform based on the Armv7 big.LITTLE architecture.
* Energy tool: The power consumption information of the Odroid platform is collected by reading directly the current sensors, while the Intel server requires the use of the Intel Performance Counter Monitor (PCM) API and its power utility to get those data
* Threads: 10 threads for each a chunk ( piece of code) is assigned
* What is/are the variables authors are observing apart of energy consumption, if any?
* CpS - time
* CpJ - time
* Watt - time
* CpJ - #Cores
* CpJ - Configuration
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* not found
* Does there exist a replication package for the reported experiments?
* not found

**ACM:**

Paper3: Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: Programming Productivity, Performance, and Energy Consumption∗

* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* Rodinia benchmark 
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware: 

Two heterogeneous single-node systems.

* Description:

Emil is a heterogeneous system that consists of two Intel Xeon E5-2695 v2 general purpose CPUs on the host, and one Intel Xeon Phi 7120P co-processing device. In total, the host is composed of 24 cores, each CPU has 12 cores that support two threads per core (known as logical cores) that amounts to a total of 48 threads. The Xeon Phi device has 61 cores running at 1.2 GHz base frequency, four hardware threads per core, which amounts to a total of 244 threads. One of the cores is used by the lightweight Linux operating system installed on the device.

Ida is a heterogeneous system that consists of two Intel Xeon E5- 2650 v4 general purpose CPUs on the host, and one GeForce GTX Titan X GPU. Similar to Emil, Ida has 24 cores and 48 threads on the host, whereas the GPU device has 24 Streaming Multiprocessors (SM), and in total 3072 CUDA cores running at base frequency of 1 GHz.

* Energy tool: x-MeterPU

Description: The use of x-MeterPU is very simple. The start() and stop() meth- ods are used to enclose code regions to be measured. The get\_value() method is used to retrieve the energy consumption (in Joules). In addition to the total energy consumption, x-MeterPU returns a log file containing all the power data with exact timestamps, which enables the production of various plots.

* Threads : Not saying
* What is/are the variables authors are observing apart of energy consumption, if any?
* Use of OpenCL, OpenMP, OpenACC, CUDA in different benchmarks from Rodinia and Spec Accel. Percentage of this frameworks used in benchmarks app
* Comparison of frameworks based on Energy and Time
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* Not mentioned
* Does there exist a replication package for the reported experiments?
* Not found

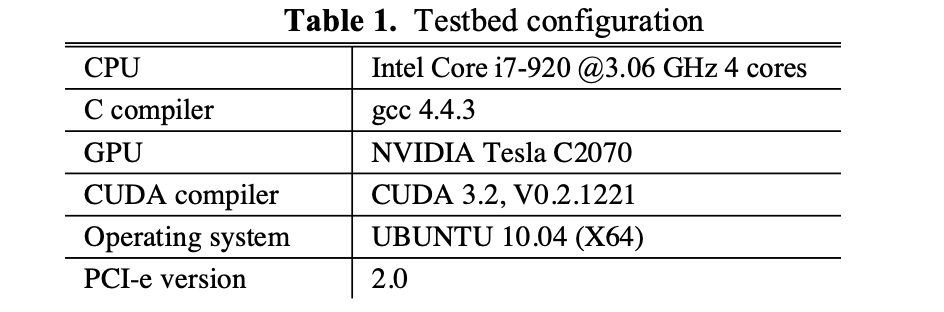
**Paper6: The Influence of the Java Collection Framework on Overall Energy Consumption**

* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* Java Collections
  + Sets: ConcurrentSkipListSet, CopyOnWriteArraySet, HashSet, LinkedHashSet, TreeSet
  + Lists: ArrayList, AttributeList, CopyOnWriteArrayList, LinkedList, RoleList, RoleUnresolvedList, Stack, Vector
  + Maps: ConcurrentHashMap, ConcurrentSkipListMap, HashMap, Hashtable, IdentityHashMap, LinkedHashMap, Properties, SimpleBindings, TreeMap, UIDefaults, WeakHashMap
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware: Linux 3.13.0-74-generic operating system, 8GB of RAM, and Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz. This system has no other software installed or running other than necessary to run this study, and the operating system daemons. Both the Java compiler and interpreter were ver- sions 1.8.0 66.
* Energy tool: RAPL, jRAPL
* Population is 25k, 250k and 1M
* How to do: github.com/greensoftwarelab/Collections-Energy-Benchmark
* What is/are the variables authors are observing apart of energy consumption, if any?
* Only time and energy for each method
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* Prior to executing a test, we ran an initial “warm-up” where we instantiated, populated (with the designated pop- size), and performed simple actions on the data structures. Each test was executed 10 times, and the average values for both the time and energetic consumption were extracted (of the specific test, and not the initial “warm-up” as to only measure the tested methods) after removing the lowest and highest 20% as to limit outliers.
* Does there exist a replication package for the reported experiments?
* Yes: github.com/greensoftwarelab/Collections-Energy-Benchmark

Where the whole contribution of actors can be found

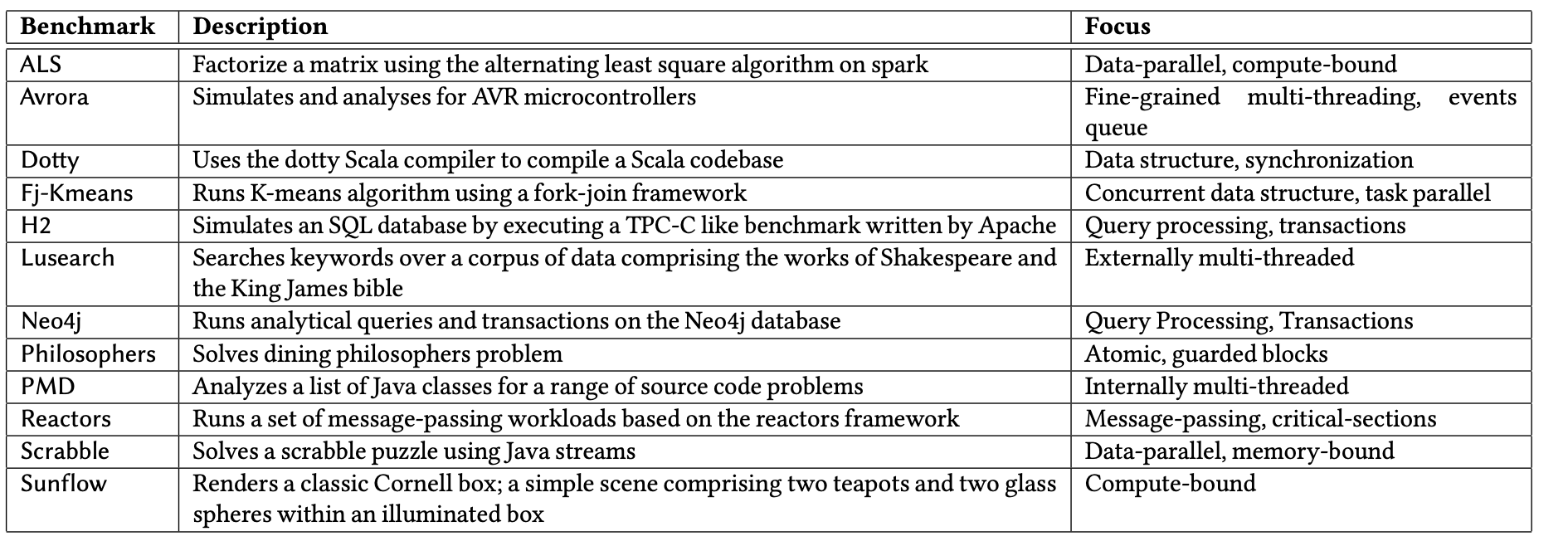
Paper18: PETRAS: Performance, Energy and Thermal Aware Resource Allocation and Scheduling for Heterogeneous Systems

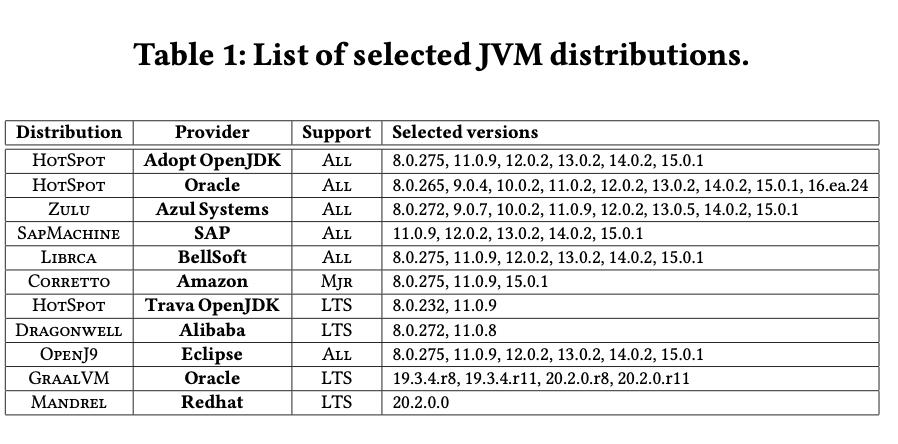
* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* Rodinia benchmark 3.0
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware:



* Energy tool: To measure peak CPU/GPU temperatures, we used the lm\_sensors application (Linux monitor- ing sensors). If a job runs on a multi-core CPU, lm\_sensors moni- tors each core’s temperature separately and we record the highest core temperature as the peak CPU temperature.
* But also kill of watt meter
* Threads: 2,4,8,16,32,64,128
* used a Linux command to change the boot arguments to disable/enable cores.
* What is/are the variables authors are observing apart of energy consumption, if any?
* Execution time over different threads
* Energy (kWh) - threads
* Peak Power (Watt) - threads
* CPU Peak Temperature - threads
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* We tested each of the schedulers by running 5000 randomly generated jobs for applications of the Rodinia benchmark and then we took the average.
* For each of the experi- ments above we ran the application 1000 times, measured the parameters and took the averages.
* Does there exist a replication package for the reported experiments?
* Not found

**Paper19: Evaluating the Impact of Java Virtual Machines on Energy Consumption**

* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* Dacapo benchmark: Avrora, H2, Lusearch, Sunflow and PMD
* Renaissance benchmark suite: ALS, Dotty, Fj-kmeans, Neo4j, Philosophers, Reaction and Scrabble
* Benchmarks used for multi-threading measurements are: Avrora and Reactors.
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware: To report on reproducible measurements, we used the cluster Dahu of the G5K platform [2] for most of our experi- ments. This cluster is composed of 32 identical compute nodes, which are equipped with 2 Intel Xeon Gold 6130 and 192 GB of RAM. Our experimental protocol enforces that the software under test is the only process executed on the node configured with a very minimal Linux Debian 9 (4.9.0 kernel version).
* Tool to measure energy: RAPL
* Remark: We note that, due to CPU energy consumption variations issues [15], we used the same node for all our experiments.
* JVM distributions used for experiment



* As no tool can accurately mon- itor the energy consumption at a thread level, we monitor the global power consumption and CPU utilization during the execution using RAPL for the energy, and several Linux tools for the CPU-utilization (htop, cpufreq).
* What is/are the variables authors are observing apart of energy consumption, if any?
* Energy consumption ratio - java version
* Average power(W) - time
* Active Software Threads Count - Time
* Summary table: Energy consumption when tuning JIT settings on HotSpot, GraalVM & J9 for different benchmarks
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* Every single experiment, therefore, reports on energy metrics obtained from at least 20 executions of 50 iterations per benchmark
* We executed 62,400 experiments by combining the 52 JVM distribu- tions with the 12 Java benchmarks, thus reasoning on 100 energy samples acquired for each of these combinations.
* Some service oriented benchmark are executed 20 min then compared in terms of energy performance
* Does there exist a replication package for the reported experiments?
* <https://github.com/chakib-belgaid/jvm-comparaison>

**Paper42: Cross-Layer Memory Management to Improve DRAM Energy Efficiency**

* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* MemBench
* DaCapo
* SciMark
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware: The experiments in this article use a Microway NumberSmasher Xeon Server machine with two sockets, each with their own Intel E5-2620v3 (Haswell-EP) processor. Each socket has six 2.4GHz cores with hyperthreading enabled (for a total of 12 hardware threads) and four 16GB DIMMs of Samsung DDR4 SDRAM with a clock speed of 2133MHz (product #: M393A2G40DB0- CPB). Each DIMM is connected to its own channel and is composed of two 8GB ranks. All experi- ments execute on an otherwise idle machine and, aside from Section 6.2, use only one socket and its local memory. We install 64-bit CentOS 6.6 and select Linux kernel version 2.6.32-431 (released in 2013) as our default operating system. Oracle’s HotSpot Java VM (build 1.9.0-ea-b76) (Paleczny et al. 2001) provides the base for our VM implementation.
* To estimate memory power and bandwidth, we employ Intel’s Performance Counter Monitor (PCM) tool (Willhalm et al. 2012). At every sample, PCM reports a variety of counter-based statistics, including the average read/write bandwidth on each memory channel (in MB/s), cache hit ratios, and an estimate (in Joules) of energy consumed by the CPU (cores and caches) and memory system.
* What is/are the variables authors are observing apart of energy consumption, if any?
* Perf(execution time) , Gb/s, DRAM, OA (GB), #Sites, S/M, #Hot, #Types, Samps/s
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* All experimental results report an average of five program runs.
* Does there exist a replication package for the reported experiments?
* Not found

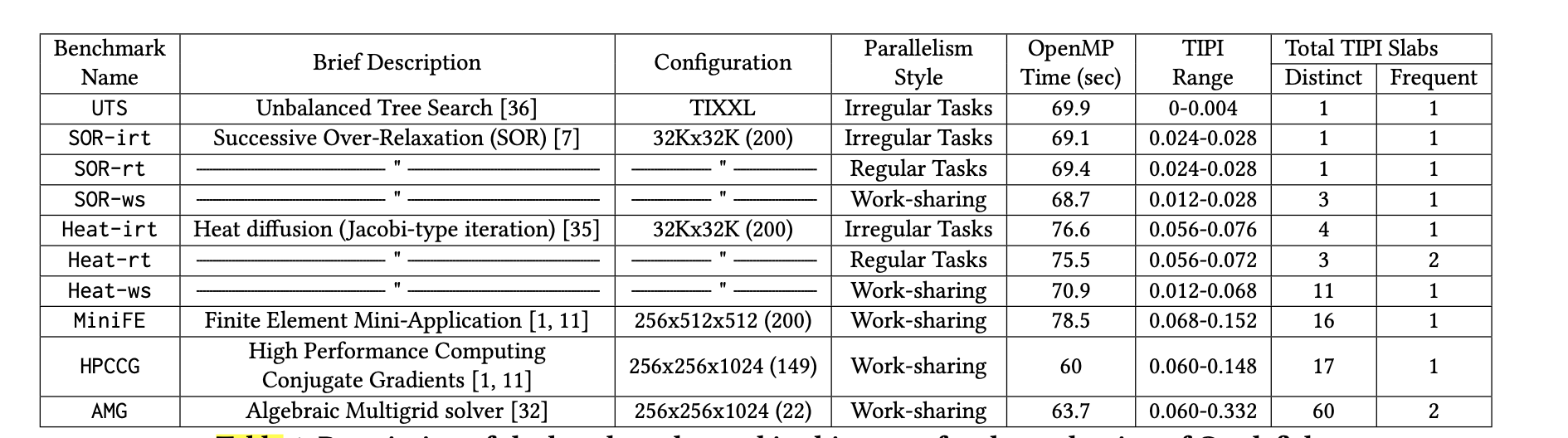
**Paper87: Approximate Communication Strategies for Energy-Efficient and High Performance NoC: Opportunities and Challenges**

**Remark: Paper is not really relevant because is based on simulation of chips architecture**

* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* COSMIC benchmark
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware (chips architecture): simulate the proposed architecture using Garnet [1] on-chip network in gem5 platform [6]. Garnet/gem5 simulation configu- rations include: instruction set architecture = ALPHA , intercon- nect Frequency = 1GHz, no. of virtual networks (VN) = 3, no. of virtual channels (VC) per VN = 4, no. of buffers per VC = 4 and XY- routing [11]
* DSENT [22] tool is integrated with gem5 to evaluate energy consumption
* ​​COSMIC benchmarks is simulated using 64-cores and routers in a 8 × 8 2D-mesh NoC. Three applications from COS- MIC are mapped in our simulations: face recognition, cifar, and reed-solomon code encoder (RS-Encoder). Face recognition (face), cifar, and RS-Encoder (RSE) application have 33, 33, and 141 tasks, respectively.
* Eight synthetic traffic patterns are simulated using 16-cores and routers in a 4 × 4 2D-mesh NoC. Traffic patterns are: Uniform Random (Uni), Tornado (Tor), Bit-complement (BitC), Bit Rotation (BitRt), Neighbor (Nbor), Shuffle (Shuf), Transpose (Tr), and Bit, Reverse (BitRv)
* What is/are the variables authors are observing apart of energy consumption, if any?
* Latency (nanoseconds) - Benchmarks
* Energy (Joules) - Benchmarks
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* Nothing mentioned
* Does there exist a replication package for the reported experiments?
* Not found

**Paper103: Cuttlefish: Library for Achieving Energy Efficiency in Multicore Parallel Programs**

**Remark: It has a high chance that experiments are done on HPC. Maybe not too useful paper**

* What are the alternatives to vacation2: i.e. what is/are the case studies used when running the experiments?
* What are the settings of the experiment (e.g. hardware, software, free variables - number of threads, and constant variables - number of cities, for example in the vacation2?
* Hardware: ​​We ran all experiments on an Intel Xeon Haswell E5-2650 v3 20- core processor with a total of 94GB of RAM. The operating system (OS) was Ubuntu 16.04.7 LTS. This processor supports core and un- core frequencies between 1.2GHz–2.3GHz and 1.2GHz–3.0GHz, re- spectively, both in steps 0.1GHz. We used MSR-SAFE [34] for saving and restoring Model-Specific-Register (MSR) values. The turboboost feature on the processor was disabled, and each benchmark used interleaved memory allocation policies supported by the numactl library.
* The HClib implementation from the official Github reposi- tory with the commit id ab310a0 is used. Both OpenMP and HClib versions of the benchmarks used the Clang compiler version 3.8.0 with the -O3 flag.
* Threads: All 20 threads used in the experiments were bound to their respective physical CPUs.
* We compared these implementations against each benchmark’s Default execution by setting the Intel power governor to performance policy. The performance power governor fixes the CPU frequency to the maximum. We chose performance power governor for Default execution as this same setting is used by several supercomputers in production
* Energy measured: RAPL
* What is/are the variables authors are observing apart of energy consumption, if any?
* Joules per instruction - execution time
* TOR Inserts per instruction - execution time
* Joules per instruction - TOR Inserts per instruction
* What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean of the median?
* Values are recorded at each 20 millisecondes
* Does there exist a replication package for the reported experiments?
* not found

**Paper123: Verified Instruction-Level Energy Consumption Measurement for NVIDIA GPUs**

**This paper have terms threads but it is only assignation of a program written in assembly language for doing the job and only on one thread the measurement is done. There is no parallelism in the benchmark. In addition the benchmark proposed is developed by the developers (even if the source is available).**