COL862
Programming Assignment-1

Submitted By: Rajesh Kedia (2014CSZ8383)

Objective:

Understand the power and energy behavior of various benchmarks on different types of x86 based systems. We explore a laptop, desktop and server machine with below configurations during our experiments.

System configuration:

Sl. No.	Parameter Name	Laptop	Desktop	Server
1	Model and build	Lenovo B5070	Dell Optiplex 9020	
2	Processor type	Intel i3 4030U	Intel i7	Xeon E5-2680
3	Processor architecture	Haswell	Haswell	Haswell
4	Max. Frequency	1.9 GHz	3.6 GHz	2.5 GHz
5	No. of sockets	1	1	2
6	No. of physical cores	2	4	24 (12 in each socket)
7	No. of virtual cores	4	8	48 (24 in each socket)
8	Main memory	2 GB	16 GB	128 GB
9	L3 cache	3 MB	8 MB	32 MB

Measurement methodology:

We used Intel's Performance Counter Monitor (PCM) tool [1] to measure the energy/power consumed by the system in a given time interval. PCM tool reports energy consumed by the socket and DRAM in the last one second. The energy consumed by the system in the last one second is also a measure of power (energy per second).

We added a new switch to the PCM tool to monitor and report energy readings only if they are above a certain threshold, else the tool quits. This feature was added so that the PCM tool runs only for the duration of the benchmark and then it quits. The data reported in the log file corresponds to the benchmark being run and requires no further filtering.

A script was written around the PCM tool which firstly runs a given number of benchmark threads and then starts the PCM instrumentation tool. The assumption is that the Operating System (OS) will handle running different threads on separate CPUs available in the socket.

Once the data is reported for each of the benchmark for various number of threads for each of them, another parsing script reads these reports and generates a summary across all benchmarks which is then read in openoffice to generate plots.

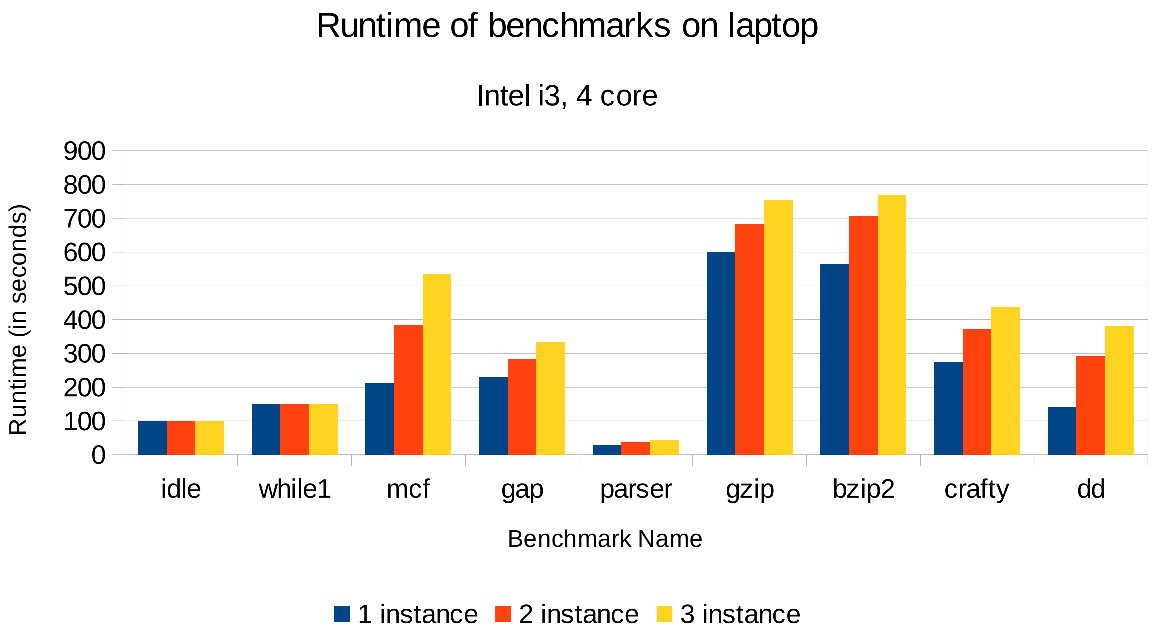
About the benchmarks:

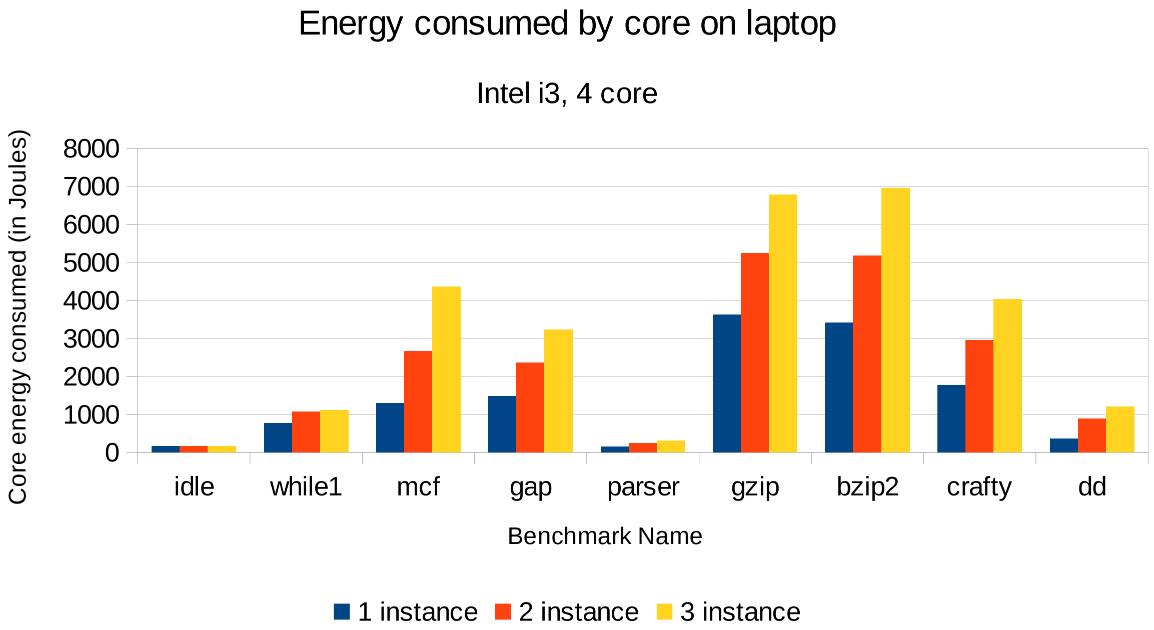
We primarily use SPEC2000 CPU benchmarks. Some of the benchmarks have been coded by us to exercise specific portions of the system.

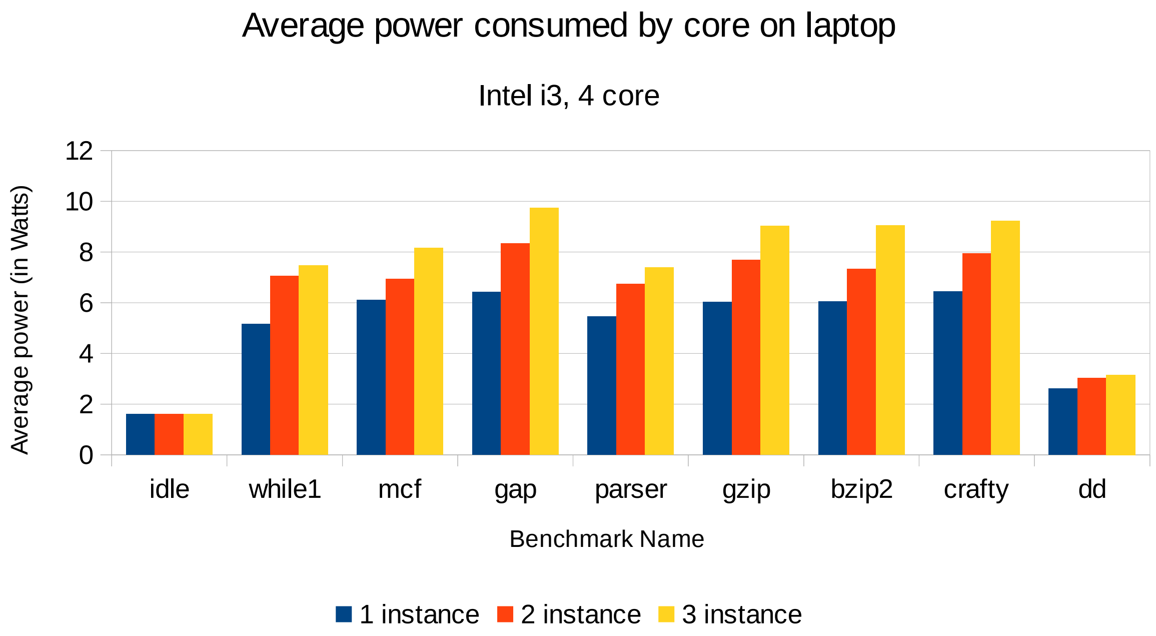
1. idle: This benchmark is with intensive application running on the system and the default processes of OS being active along with a terminal and the power measurement process.
2. while1: It is a simple while loop which runs for a given number of iterations. There is no data access involved in the code and it is primarily intended to keep CPU busy and active.
3. mcf: [2] SPEC 2000 benchmark. A benchmark derived from a program used for single-depot vehicle scheduling in public mass transportation. The program uses around 190MB for 64-bit systems and is used to stress the cache and memory subsystem.
4. gap: [3] It implements a language and library designed mostly for computing in groups (GAP is an acronym for Groups, Algorithms and Programming).
5. parser: [4] The Link Grammar Parser is a syntactic parser of English, based on link grammar, an original theory of English syntax.
6. gzip: [5] gzip (GNU zip) is a popular data compression program based on LZ77 encoding. gzip's reference workload has five components: a large TIFF image, a webserver log, a program binary, random data, and a source tar file. Each input set is compressed and decompressed at several different blocking factors ("compression levels"), with the end result of the process being compared to the original data after each step. No file I/O, all compression and decompression happens in memory.
7. bzip2: [6] bzip2 benchmark is based on the compression algorithm. 256.bzip2's reference workload has three components: a large TIFF image, a program binary, and a source tar file.
8. crafty: [7] Game playing program (plays chess). It is primarily an integer code, with a significant number of logical operations such as and, or, exclusive or and shift. It can be configured to run a reproducible set of searches to compare the integer/branch prediction/pipe-lining facilities of a processor. The reference input solves 5 different chess board layouts, with varying "depths" to which it will search the tree of possible moves, for the next move.
9. dd: This benchmark uses the Linux dd command to copy a 4.5GB file from one location to another in the disk. In case of parallel threads, the input file is same while output file names are different. We use dd -if “pathname” -of “pathname”.

Results:

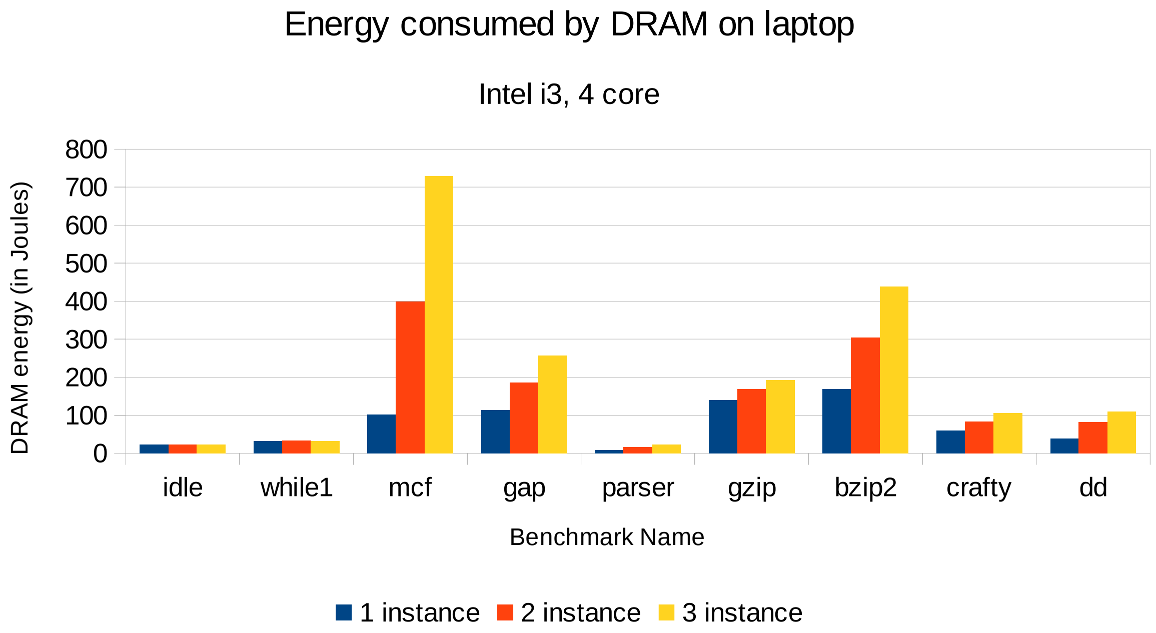
We extract and plot 5 different characteristics of the benchmarks for each of the target system. Firstly we plot the total runtime of the benchmark, then we plot the total CPU energy consumed by the benchmark, followed by the average CPU power consumed by the benchmark. Subsequently, we plot total DRAM energy consumed and the average DRAM power consumed by the benchmark.


Runtime of benchmarks on laptop
Intel i3, 4 core
Runtime (in seconds)
900
800
700
600
500
400
300
200
100
0
idle
while1
mcf
gap
parser
gzip
bzip2
crafty
dd
Benchmark Name
1 instance
2 instance
3 instance

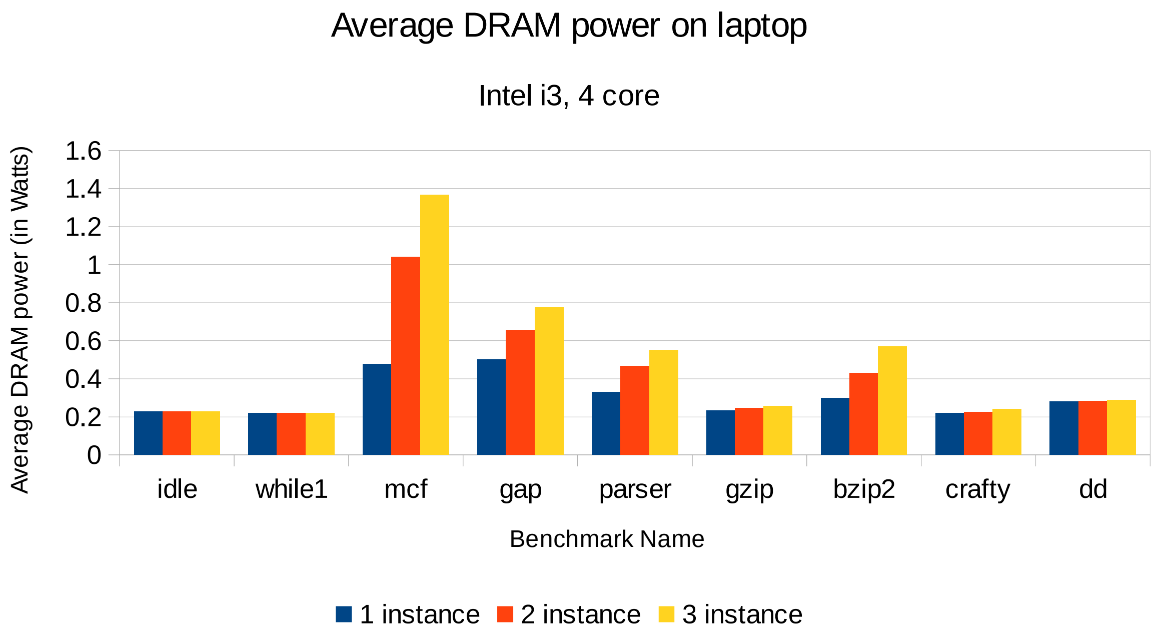

Energy consumed by core on laptop
Intel i3, 4 core
Core energy consumed (in Joules)
8000
7000
6000
5000
4000
3000
2000
1000
0
idle
while1
mcf
gap
parser
gzip
bzip2
crafty
dd
Benchmark Name
1 instance
2 instance
3 instance


Average power consumed by core on laptop

Intel i3, 4 core

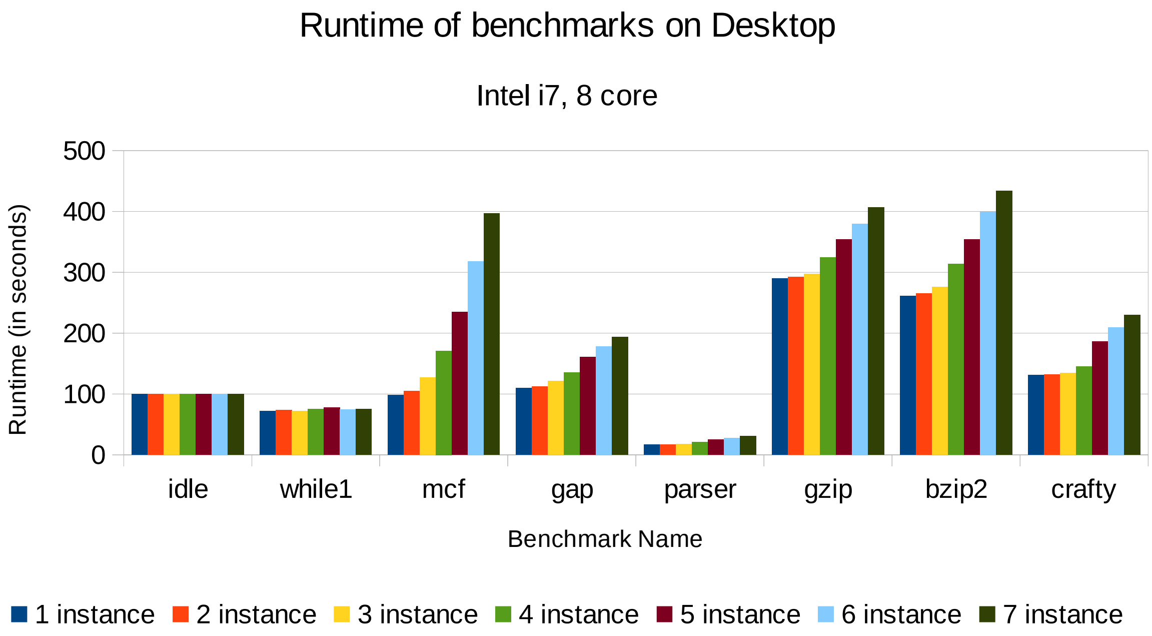

Energy consumed by DRAM on laptop

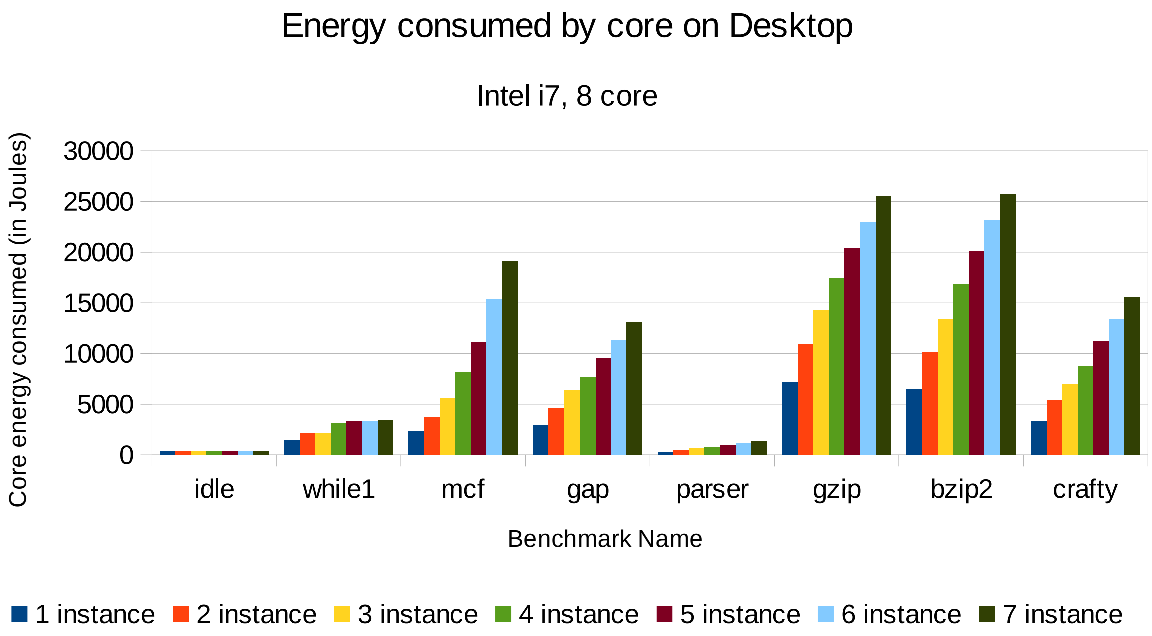
Intel i3, 4 core

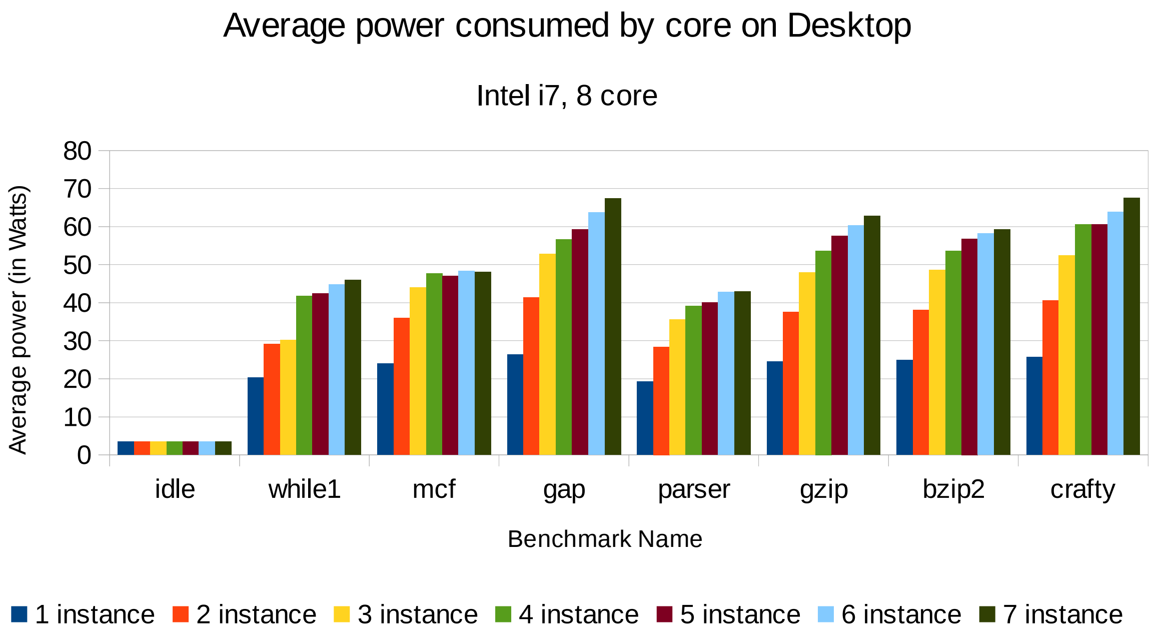


Analysis:

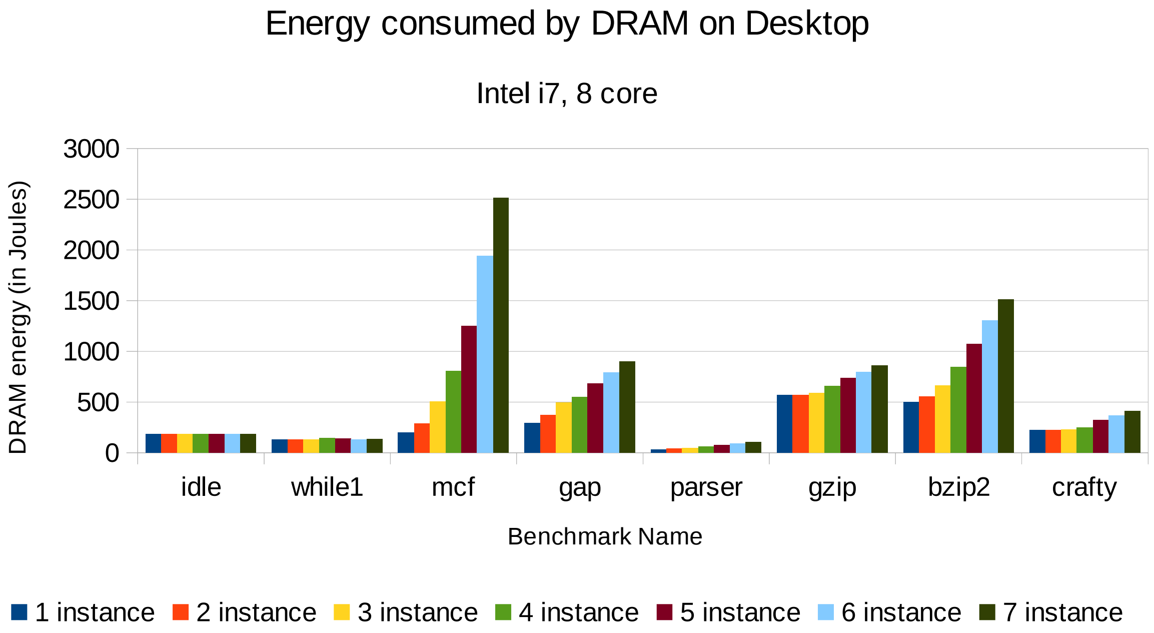
1. For while1, we see that invoking multiple number of threads has no impact on the runtime since each of the threads can run independently and do not cause contention in shared resources like LLC or memory. Benchmarks like mcf which are intensive on cache and memory access show proportional increase in the runtime with increase in the number of threads due to contention for memory. This is further confirmed with the significant increase in the DRAM access power for the mcf benchmark.
2. For most of the benchmarks, we see that the total energy consumed in running one instance of the benchmark versus running multiple instance is not proportional to the number of instances. Total energy consumed for running multiple instances is a fractional increase over single instance, thus implying that it is more energy efficient to do more jobs in parallel using a multi-core machine and then go for longer sleep. This follows the claim of authors in [8].
3. We could also identify certain behavior of benchmarks based on above results, without going into the details of benchmarks. bzip2 shows a larger impact due to running of multiple threads compared to gzip. Thus we can infer that bzip2 is more memory intensive than gzip.
4. Though dd is memory intensive, the DRAM power and CPU power remains almost constant for across multiple instances of the benchmark. This is because it stresses the disk subsystem which is comparatively very slow and hence is the bottleneck.


Runtime of benchmarks on Desktop
Intel i7, 8 core
Runtime (in seconds)
500
400
300
200
100
0
idle
while1
mcf
gap
parser
gzip
bzip2
crafty
Benchmark Name
1 instance
2 instance
3 instance
4 instance
5 instance
6 instance
7 instance

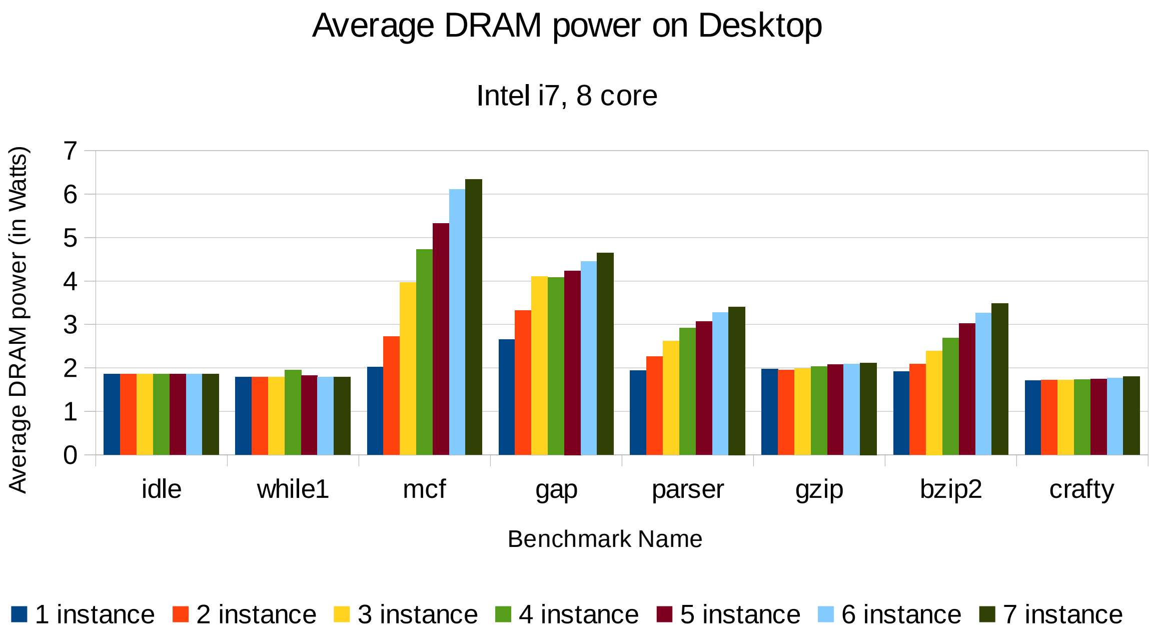

Energy consumed by core on Desktop
Intel i7, 8 core
Core energy consumed (in Joules)
30000
25000
20000
15000
10000
5000
0
idle
while1
mcf
gap
parser
gzip
bzip2
crafty
Benchmark Name
1 instance
2 instance
3 instance
4 instance
5 instance
6 instance
7 instance



Average power consumed by core on Desktop
Intel i7, 8 core
Average power (in Watts)
0
10
20
30
40
50
60
70
80
idle
while1
mcf
gap
parser
gzip
bzip2
crafty
Benchmark Name
1 instance
2 instance
3 instance
4 instance
5 instance
6 instance
7 instance

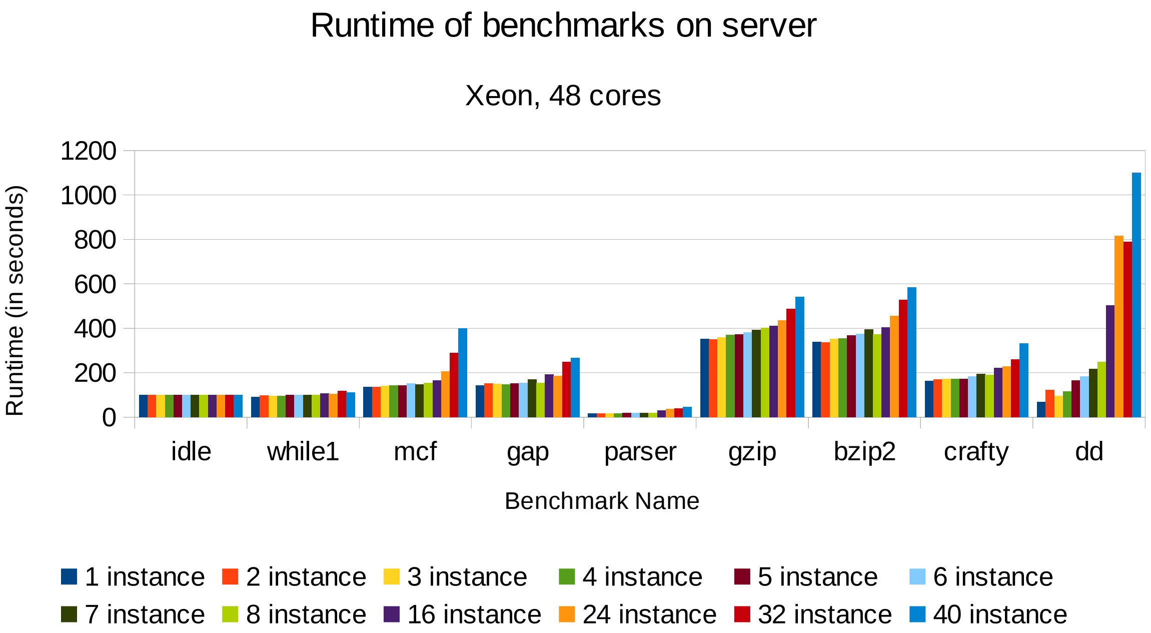


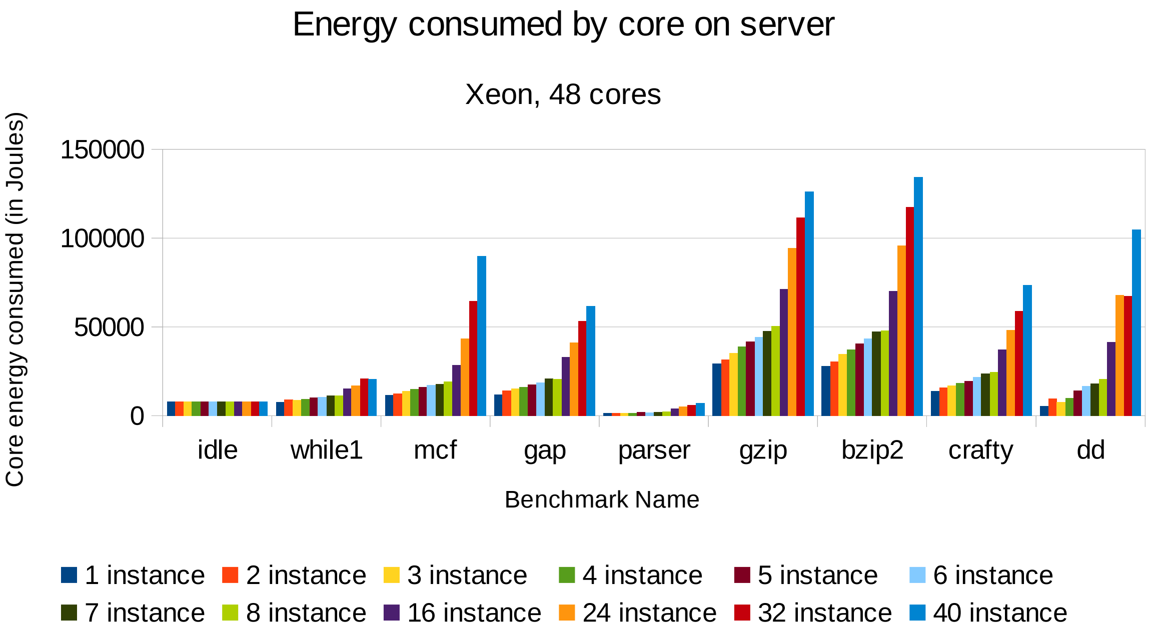
Energy consumed by DRAM on Desktop
Intel i7, 8 core
DRAM energy (in Joules)
0
500
1000
1500
2000
2500
3000
idle
while1
mcf
gap
parser
gzip
bzip2
crafty
Benchmark Name
1 instance
2 instance
3 instance
4 instance
5 instance
6 instance
7 instance

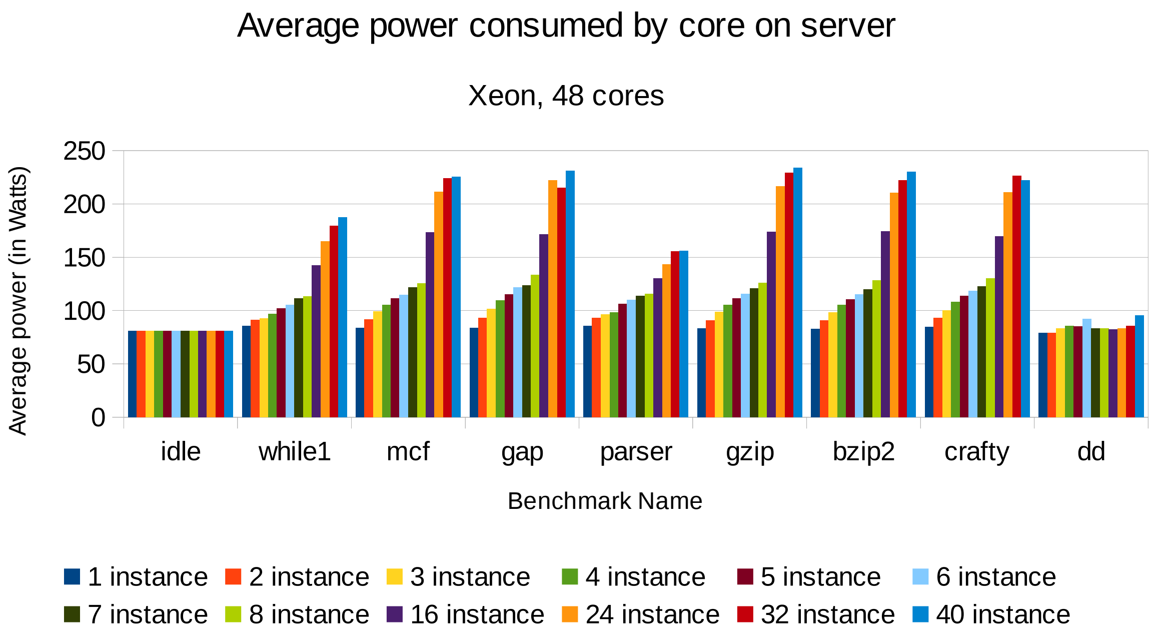


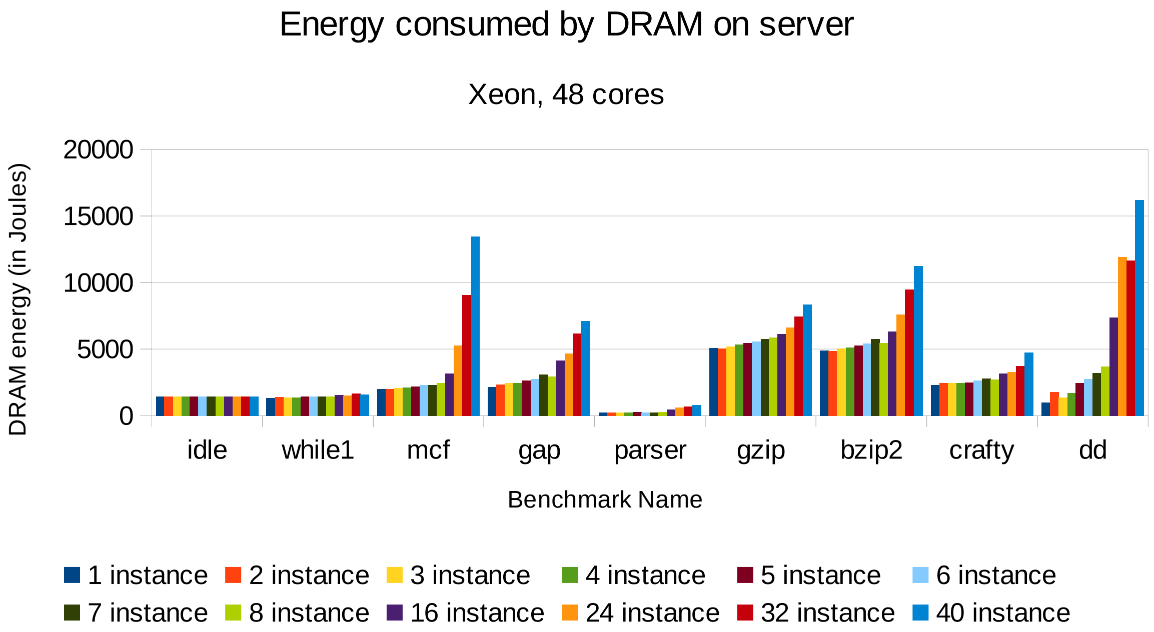
Analysis for Desktop results:

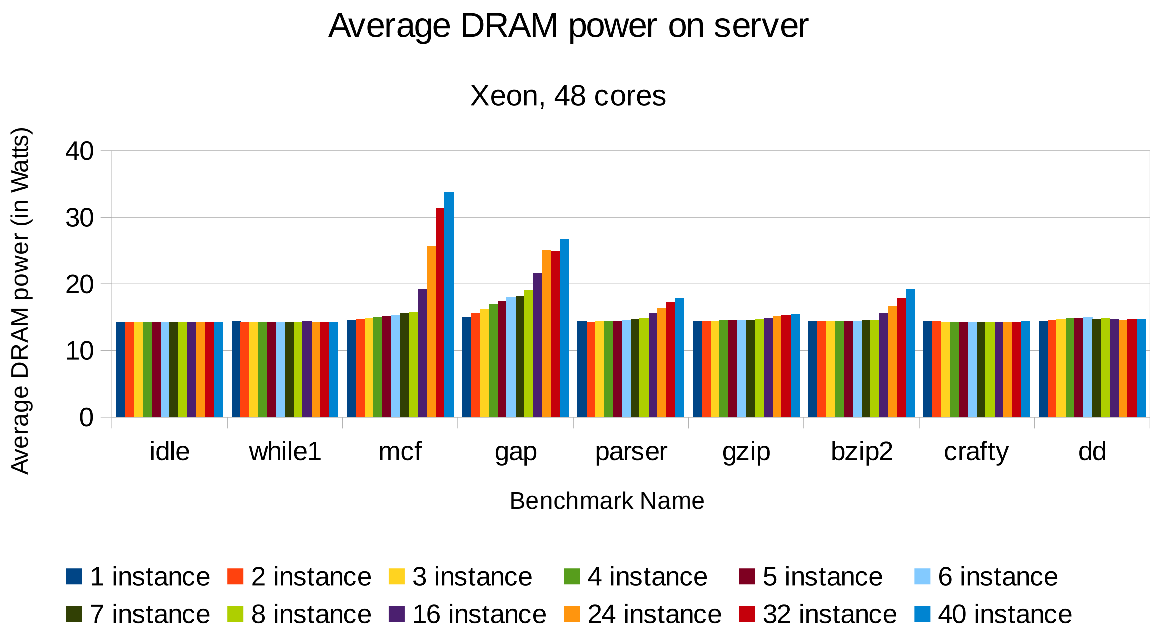
1. We see similar trends in the results for desktop compared to laptop.
2. We couldn't run the dd benchmark on the desktop due to the disk having some defects leading to problems in stressing the disk.
3. Some of the benchmarks like gzip do not show an increase in DRAM power until 4 threads, but then have a small effect due to more number of threads. This is due to contention in L3 with more number of threads.
4. DRAM power shows lower increase than idle power (compared to laptop) due to presence of larger cache on Desktop and the base power being higher.
5. We also observe that the default scheduling settings on Ubuntu is not aware of SMT. The threads are allocated on same or different physical cores in different invocation of the threads. Ideally for performance reasons, one would expect different physical cores being allocated if number of threads are less than number of physical cores, but it was observed not to happen always.


Runtime of benchmarks on server
Xeon, 48 cores
Runtime (in seconds)
1200
1000
800
600
400
200
0
idle
while1
mcf
gap
parser
gzip
bzip2
crafty
dd
Benchmark Name
1 instance
2 instance
3 instance
4 instance
5 instance
6 instance
7 instance
8 instance
16 instance
24 instance
32 instance
40 instance


Energy consumed by core on server
Xeon, 48 cores
Core energy consumed (in Joules)
150000
100000
50000
0
idle
while1
mcf
gap
parser
gzip
bzip2
crafty
dd
Benchmark Name
1 instance
2 instance
3 instance
4 instance
5 instance
6 instance
7 instance
8 instance
16 instance
24 instance
32 instance
40 instance


Average power consumed by core on server
Xeon, 48 cores
Average power (in Watts)
250
200
150
100
50
0
idle
while1
mcf
gap
parser
gzip
bzip2
crafty
dd
Benchmark Name
1 instance
2 instance
3 instance
4 instance
5 instance
6 instance
7 instance
8 instance
16 instance
24 instance
32 instance
40 instance

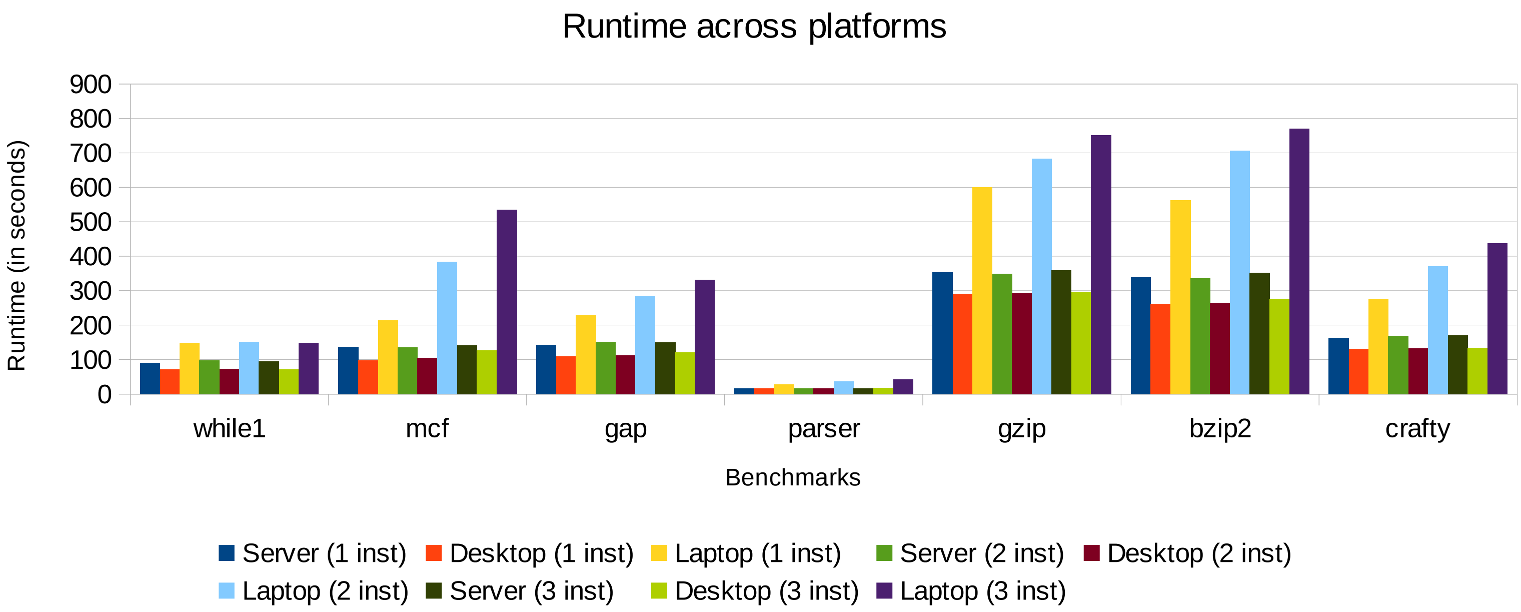

Energy consumed by DRAM on server
Xeon, 48 cores
DRAM energy (in Joules)
20000
15000
10000
5000
0
idle
while1
mcf
gap
parser
gzip
bzip2
crafty
dd
Benchmark Name
1 instance
2 instance
3 instance
4 instance
5 instance
6 instance
7 instance
8 instance
16 instance
24 instance
32 instance
40 instance

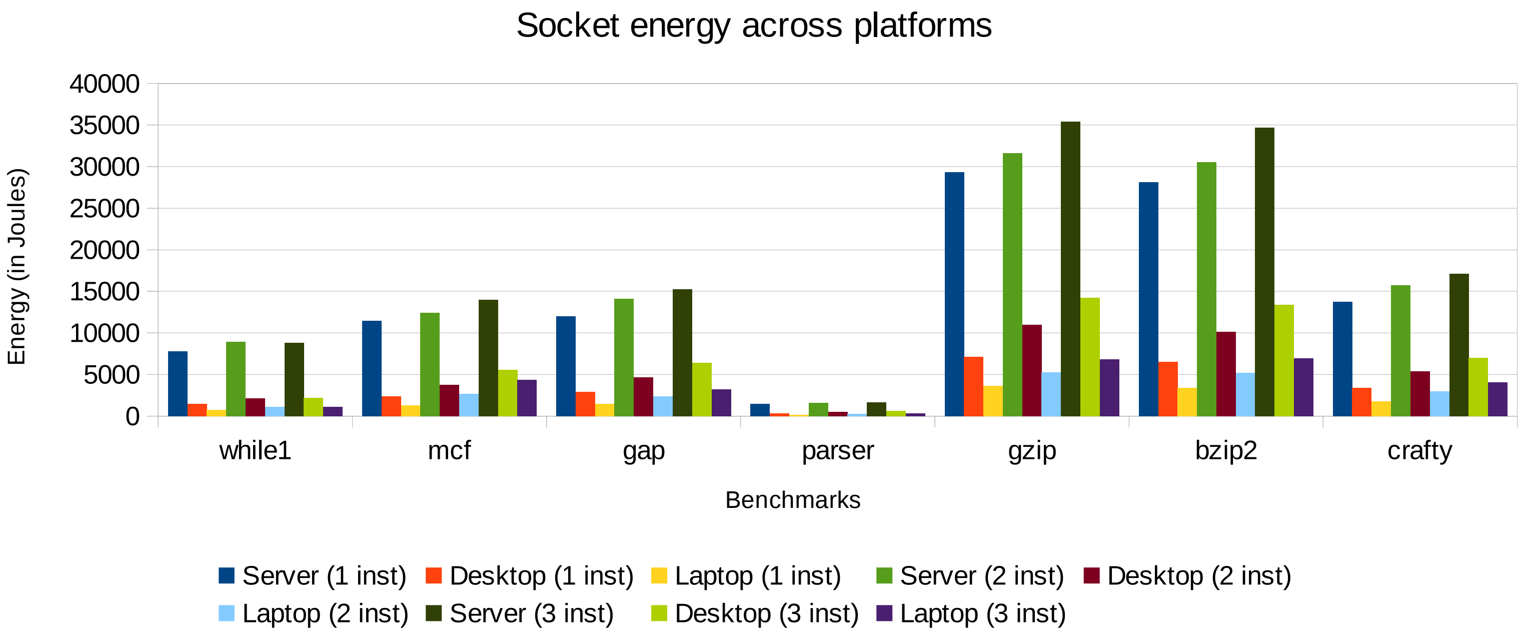


Analysis for results for server:

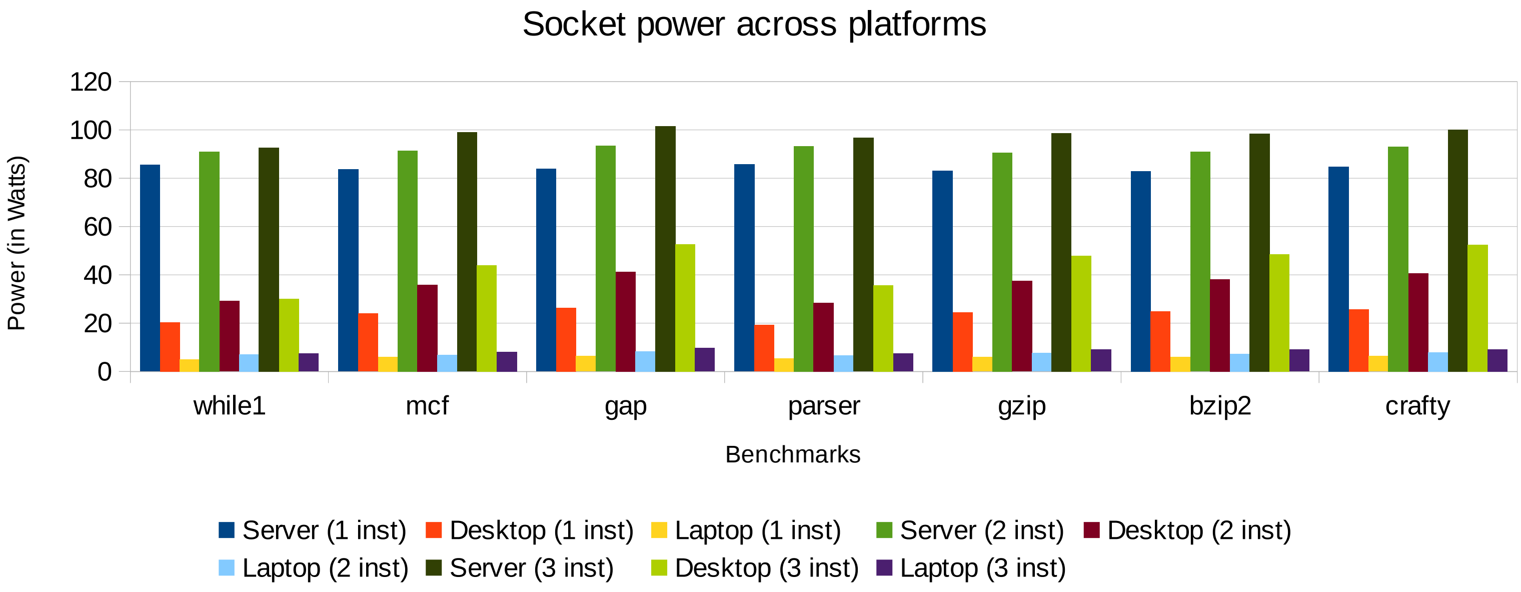
1. We do not see much impact on DRAM power until 8 threads of the application. This is because of the high base DRAM power compared to other systems. The DRAM power plot is flat for until 7 instances for most of the benchmarks including mcf. For most of the other applications, the DRAM power is close that in idle benchmark.
2. Most of the other trends seen in laptop and desktop for these benchmarks hold true for server as well.
3. We see that dd benchmark retains the same average DRAM and CPU power for even 40 threads due to the disk being the bottleneck.

Comparison of results across different platforms:

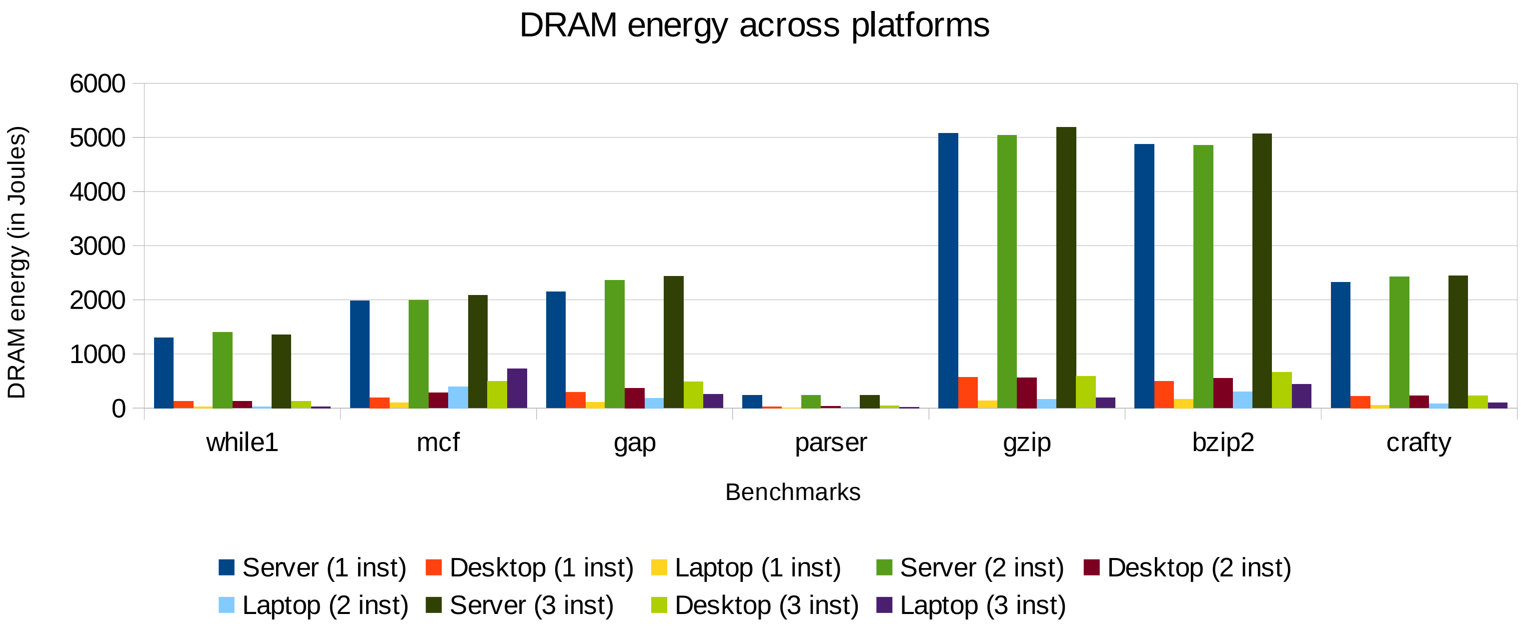


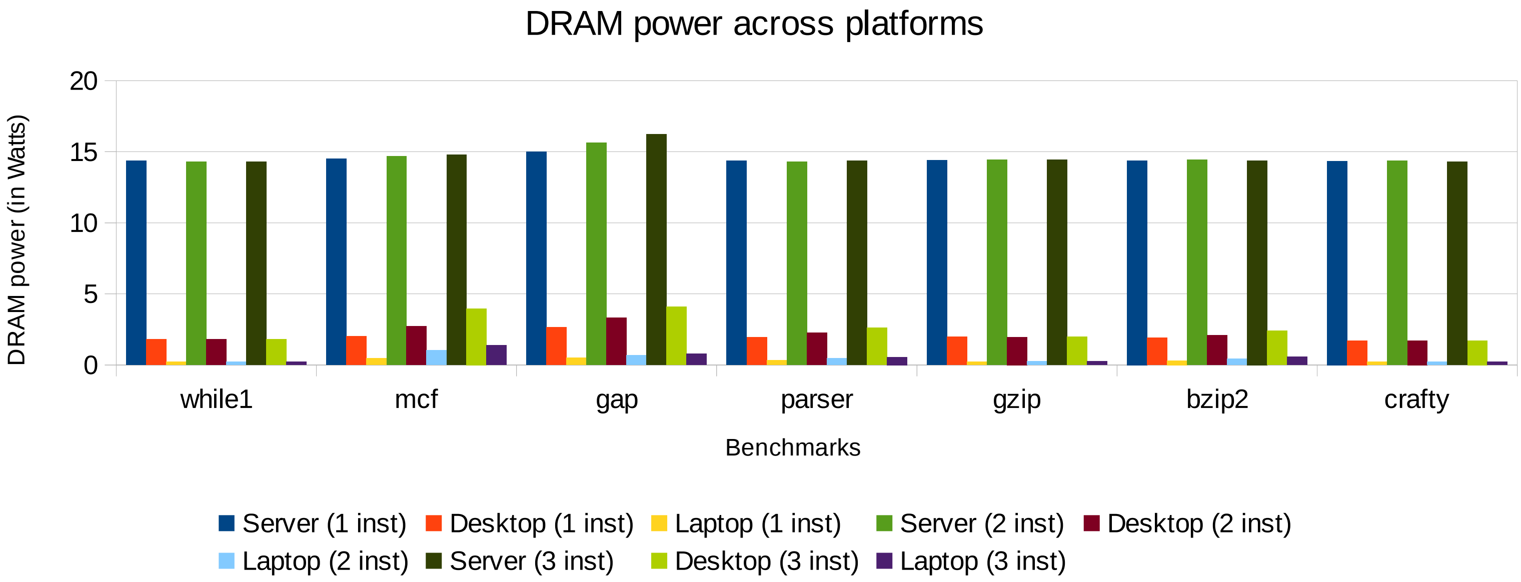


Socket energy across platforms
Energy (in Joules)
40000
35000
30000
25000
20000
15000
10000
5000
0
while1
mcf
gap
parser
gzip
bzip2
crafty
Benchmarks
Server (1 inst)
Desktop (1 inst)
Laptop (1 inst)
Server (2 inst)
Desktop (2 inst)
Laptop (2 inst)
Server (3 inst)
Desktop (3 inst)
Laptop (3 inst)



Socket power across platforms
Power (in Watts)
120
100
80
60
40
20
0
while1
mcf
gap
parser
gzip
bzip2
crafty
Benchmarks
Server (1 inst)
Desktop (1 inst)
Laptop (1 inst)
Server (2 inst)
Desktop (2 inst)
Laptop (2 inst)
Server (3 inst)
Desktop (3 inst)
Laptop (3 inst)





Analysis of the results:

1. We observe that across benchmarks and number of threads, the runtime for the server is higher than the runtime on the Desktop. One possible reason for this is the difference in the max. frequency of the processors. The Desktop can run at 3.6GHz while the server is limited to 2.5GHz.
2. CPU energy consumed for doing the same task is highest for server, followed by Desktop and then the laptop. Thus, if performance is not critical, using laptop is the most energy efficient means of computation for these benchmarks.
3. For DRAM energy, we see that laptop shows a higher energy consumption compared to Desktop (for multiple mcf threads). This is primarily because of the larger size of cache memory present on the Desktop causing lesser stress on DRAM compared to laptop platform.
4. Idle DRAM power for server is ~7 times of the Desktop, which closely matches the size of them as well (server has 8x the DRAM available in Desktop). Similarly, the DRAM power of Desktop is ~7.5 times of the laptop which follows their size difference (Desktop DRAM

is 8x the size of laptop DRAM). However this trend changes for memory intensive benchmarks like mcf where the smaller DRAM systems suffer more cache misses and hence more accesses to memory, overriding the static effects of memory size. It is ~3x DRAM power from laptop to Desktop or Desktop to server.

References:

[1] Intel PCM tool. https://software.intel.com/en-us/articles/intel-performance-counter-monitor
[2] SPEC2000 MCF benchmark
https://www.spec.org/cpu2000/CINT2000/181.mcf/docs/181.mcf.html
[3] SPEC2000 GAP benchmark
https://www.spec.org/cpu2000/CINT2000/254.gap/docs/254.gap.html
[4] SPEC2000 parser benchmark
https://www.spec.org/cpu2000/CINT2000/197.parser/docs/197.parser.html
[5] SPEC2000 gzip benchmark
https://www.spec.org/cpu2000/CINT2000/164.gzip/docs/164.gzip.html
[6] SPEC2000 bzip2 benchmark
https://www.spec.org/cpu2000/CINT2000/256.bzip2/docs/256.bzip2.html
[7] SPEC2000 crafty benchmark
https://www.spec.org/cpu2000/CINT2000/186.crafty/docs/186.crafty.html
[8] Meng Zhu and Kai Shen, Energy Discounted Computing on Multicore Smartphones. 2016 USENIX Annual Technical Conference (USENIX ATC 16).
https://www.usenix.org/conference/atc16/technical-sessions/presentation/zhu