Heterogeneity By The Numbers
A Study of the ODROID XU+E big.LITTLE Platform

Marcus Hähnel, Hermann Härtig
Operating Systems Group
Technische Universität Dresden
{*mhaehnel,haertig*}*@tudos.org*

Abstract

The recent availability of modern big.LITTLE ARM chips featuring heterogeneous processor cores has enabled a practical investigation of the heterogeneous chip concept and its implications for performance and energy. The ODROID XU+E platform produced by Hardkernel integrates hardware power monitors for the individual chip clusters, GPU and Memory. In this paper we investigate the detailed energy characteristics of the hardware components and applications to highlight the platform's potential for energy-aware resource management of the platform. We measure network, storage and CPU energy consumption as well as parallel application performance. We find that the platform offers interesting sweet-spots for fine tuning of resource scheduling, but also poses new challenges, for example for quantifying the cost of switching between clusters.

1 Introduction

The ODROID XU platform is available sice 4/2013. It is one of the first developer boards to feature the Samsung Exynos5 Octa 5410 and is an open architecture for developers to explore the capabilities of the ARM big.LITTLE platform. The board itself features many connectivity options as well as different storage interfaces and can run on Linux and Android. While previous work [8, 5] has looked at the potential of big.LITTLE architectures there exists, to the best of our knowledge, no work characterizing a complete, commercially available platform. Especially in the face of current work on Energy/Utility aware scheduling [7] trade-offs, as presented by these novel platforms, play an important role in modeling and scheduling future, highly adaptive heterogeneous systems. This becomes more important as heterogeneity and specialization will increase in the future due to an increasing amount of dark silicon in modern process nodes [4] and the resulting increase in in specialized components. To provide a sound basis for future energy related research of heterogeneous architectures we perform a detailed analysis of the energy behavior of this platform's software and hardware. We characterize the different components in terms of Energy/Utility – the trade-off between a resource's provided performance and its energy consumption.

After describing our hardware and measurement setup in Section 2 we analyze the network, storage, applications and switching energy characteristics in Sections 3-6 before we conclude our work, outline future research and the possibilities of the platform in the larger framework of Energy/Utility in Section 7.

2 Hardware Platform and Measurement Setup

The ODROID XU[1], produced by Hardkernel[2] uses a Samsung Exynos5 Octa 5410 processor which consists of two clusters of 4 CPU cores each. The *big* cluster uses Cortex™-A15 cores while the *LITTLE* cluster uses Cortex™-A7 cores. All cores have 32 kB of data cache and an equally sized instruction cache. The A15 cluster also has 2 MB of ECC protected L2 cache. The A7 cluster features 512 kB of unprotected L2 cache. The *big* cores run in a frequency range of 800 MHz to 1600 MHz, the *LITTLE* cores run between 500 MHz and 1200 MHz. Due to problems in the design of the cache coherent interconnect of the Exynos 5410 the platform does not feature cache coherency between the clusters [1]. Because of this big.LITTLE is implemented using a technique called *cluster switching* where only either the *big* or the *LITTLE* cores can be active at the same time.

The board features 2 GB LPDDR3 memory on package. For storage an on-board microSD slot connected via SD3.0 and an eMMC 4.5 connector for the processors eMMC controller are provided. Our 16 GB eMMC module contained the operating system and was also used to

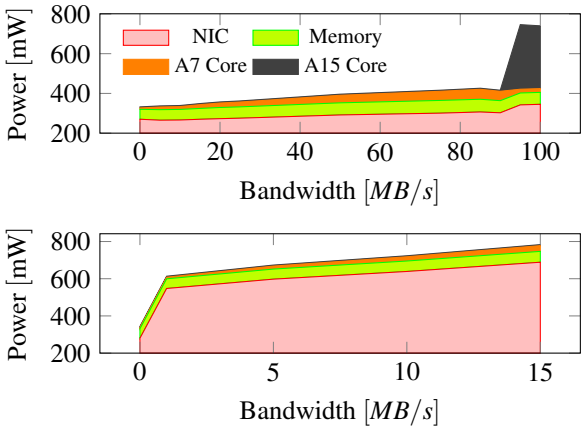

Figure 1: Power for sending over Ethernet (top) and WLAN (bottom)

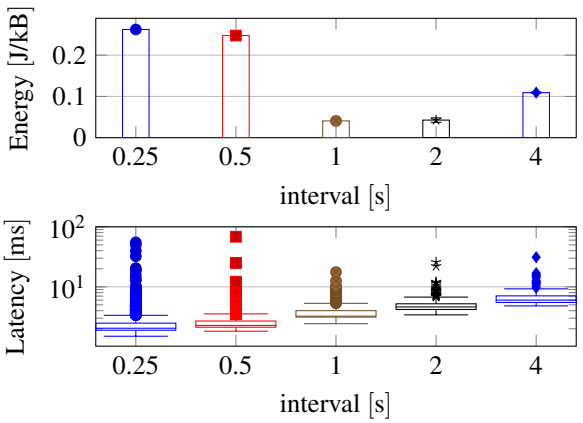

Figure 2: Energy and Latency (box plot) implications of message interval over WLAN

host the benchmarks evaluated in this paper. The microSD card was only used as target medium for storage benchmarks. The Ethernet chip that is on-board is connected via an USB 2.0 HSIC interface.

The platform contains four INA231[3] current sensors. They can measure the power consumption of the *big* and *LITTLE* cluster, the GPU and the memory individually. We further used an external ODROID *Smart Power* supply that can provide readings of the externally supplied power to a measurement computer.

Benchmarks were performed using our dedicated ODROIDReader[7] tool. This tool repeatedly executes benchmarks using different environments (such as frequencies, *big/LITTLE* configurations etc.) while measuring all available metrics that are directly related to energy such as voltages of the different regulators, CPU frequency, temperatures of the cores, power, current and voltage values supplied by the power monitors, fan speed of the CPU fan, as well as the power, energy and current values of the external power supply. All values were sampled at frequency of at least 1 Hz. Some that showed fast changes in energy usage, such as the switching benchmarks, were sampled at 5 Hz.

3 Network Stack

Our first target for evaluation was the network stack, as we wanted to evaluate the applicability of our previous energy management approach of energy-efficient network bonding [6] for the platform. The 100 Mbit/s Ethernet chip used was the on-board LAN9730[4] while the Wireless LAN was an ODROID WiFi Module 3 designed for the board which supports IEEE 802.11b/g/n networks and uses a Realtek RTL8188CUS-GR chip.

We measured the energy consumption at different bandwidths using benchcat[5] as a benchmarking tool. The wireless was only tested in g mode because an n-router was not available at the time of the benchmark. The board was positioned 0.1 m from the access point. The numbers for both network devices can be seen in Figure 1, broken down into the different consumers. The frequency and cluster selection were set to ondemand. The board's static energy consumption of about 1.8 W is not included in the graphs. The energy consumption for high bandwidth Ethernet transfers is mostly driven by costs in the network stack. This also leads to the activation of the *big* cores at high bandwidths. For the wireless network energy is mostly driven by the consumption of energy in the hardware. Due to the low achieved bandwidths the network stack is no energy driver in this scenario.

The WLAN also had an interesting tail power effect, not found to that degree in the Ethernet (probably due to less sophisticated power management). A similar effect has been described by Pathak et al. [10]. We investigated the influence of the interval of sending data on the power consumption. For this we sent ping messages with 1KB/s of data in intervals of 0.25, 0.5, 0.75, 1, 2 and 4 seconds, with individual payload sizes chosen such that the desired data rate was matched. We repeated the experiment 10 times for each interval, each measurement ran one minute. The distribution of the energy measurements of the 10 runs as well as the observed latencies of all 600 samples can be seen in Figure 2.

While the latency increases with increasing send intervals due to the hardware entering deeper power-saving states the energy consumption is lowest at 1 to 2 second intervals. At shorter intervals more energy is spent because the wireless cannot sleep, at longer intervals the increased package size increases the required send energy. When positioning the router farther away from the ODROID the power consumption increased by 60 mW for 2.5 m and by 180 mW for 5 m of distance.

4 Storage

We evaluated the performance of the different storage options using a 4 GB Class-10 microSD card and a 16 GB eMMC 4.5 module in their respective slots. While we already suspected the energy consumption of the microSD card to be worse than that of the eMMC storage we wanted to quantify how much of a difference the used storage medium makes. For this we used the `DBENCH`[6] benchmark that plays back a trace of disk I/O operations in its local operation mode without network involvement. The benchmark was set to replay the client.txt network profile exactly once.

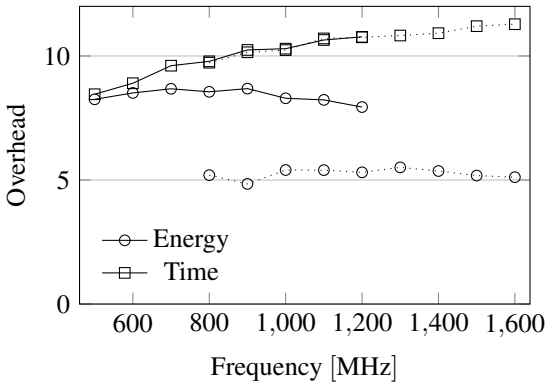

Figure 3: Energy overhead of SD card over eMMC, `DBENCH`. Dotted denotes the *big*, solid the *LITTLE* core

Figure 3 shows that the overhead of using the microSD card over the eMMC was astonishingly high, both in terms of energy as well as time, with the benchmark running about nine times as long on microSD as on eMMC and requiring more than ten times the amount of energy in the worst cases. Energy overhead is for dynamic energy. While the runtime did not vary with increasing frequency for the SD card (a nearly constant 2160 seconds) because I/O was the bottleneck it did change from 256 s to 192 s for the eMMC card with increasing frequency. This can also be seen in Figure 3 as an increase in time overhead.

The energy overhead was reduced when running at higher frequencies, because the fast eMMC caused the file system stack to make use of the faster frequencies and not be hampered by I/O latency. While this caused a reduction in runtime for the eMMC case it also causes an increase of energy usage (Compare Figure 4). The SD card does not benefit from faster frequencies but, at the same time, also keeps the power consumption of the processor low.

The system's power draw (incl. static power) ranged between 2.1 W and 3.89 W on eMMC and 2.1 W - 2.8 W when on microSD. This aspect is interesting if you want to schedule your resources under the consideration of

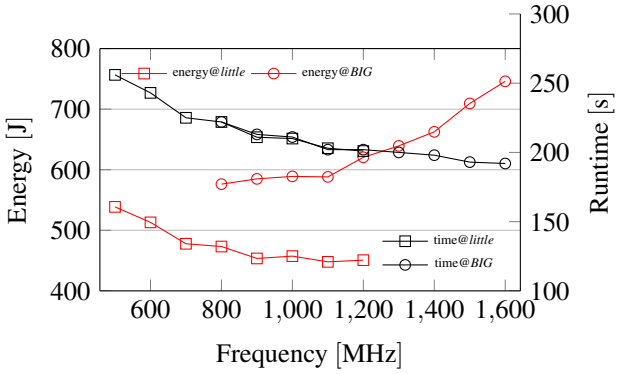

Figure 4: Energy cost versus runtime of `DBENCH` on eMMC 4.5

peak current draw to increase effective battery capacity as mandated by Peukert's Law [9] and investigated by Aksanli [2].

The total energy cost and runtime of `DBENCH` on eMMC is illustrated in Figure 4. Squares denote running on the *LITTLE* core, circles denote running on the *big* core. The graph shows that there is no time benefit to running on *big* when considering frequencies both cores can provide. The runtime does not differ between the clusters for equal frequency while the energy consumption increases significantly for the *big* cluster.

5 Applications

To evaluate the difference between executing applications on each of the clusters we used the NAS Parallel Benchmarks (NPB) [3] which we compiled for Android with OpenMP support. We ran individual benchmarks at the different frequencies of the cores, always using all 4 available cores.

Figure 5 shows the results broken down into the components that we were able to isolate.The individual groups are the benchmarks at different sizes. The sizes were chosen sucht that the average runtime of the benchmark was close to 10 minutes, with the minimum runtime at least one minute. This was required because we had to run each of the 11 benchmarks at all 16 available frequencies. We also waited for a minute to between benchmarks to let the CPU cool down again and to allow the fan to stop spinning. Each benchmark was repeated 4 times to isolate outliers. The usual standard deviation of the energy values between the runs was below 1 % of the measured value, with the maximum standard deviation at 2.1, 3.9, 4.8, 4.1 and 3.8 % of the measured values for A7, A15, Memory, GPU and external power respectively.

To keep the figure readable we chose to omit one benchmark (CG.B) which we ran but which would not contribute any new information to the figure. We also

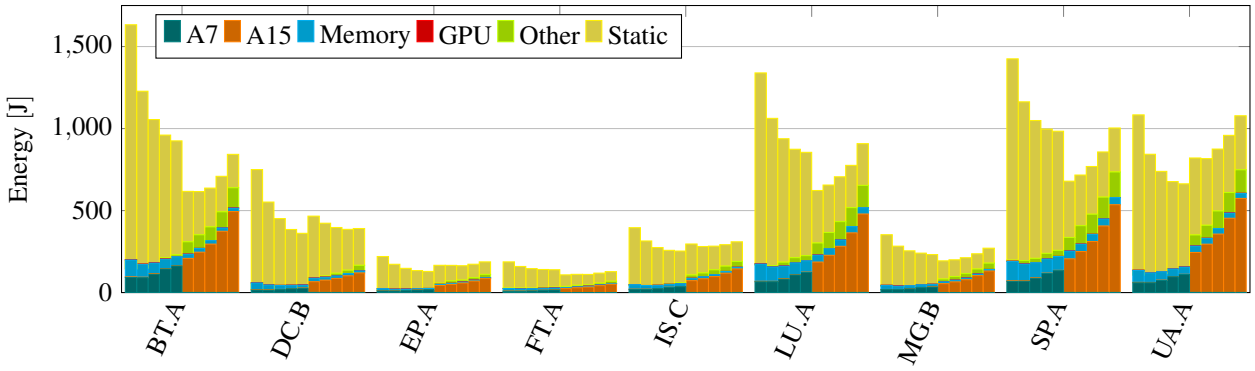

Figure 5: NPB at frequencies (left to right) 0.5, 0.7, 0.9, 1.1 & 1.2 GHz on *LITTLE* and 0.8, 1, 1.2, 1.4 & 1.6 GHz on the *big* cluster.

show only a representative subset of the frequencies due to space constraints. The complete measurement data, including the raw data and the program to interprete it are available on our GitHub[7].

As can be seen there are different sweet spot frequencies (and sweet-spot clusters) for the individual benchmarks. For example SP.A seems to benefit significantly from the out of order pipeline or the larger cache of the *big* core, while UA.A performs significantly better from an energy standpoint on the *LITTLE* core (unless it is run at the lowest frequency). The UA benchmark is mainly memory bound with irregular, continuously changing memory access patterns which cannot benefit significantly from the larger cache of the *big* cores due to this access pattern. SP, in contrast is a pentadiagonal PDE solver that is quite memory heavy and can benefit from the larger cache and the out-of-order pipeline to limit stalls.

All this is only valid when taking the static energy consumption into account. As soon as the system has to be powered anyways it is almost always cheaper energy-wise to use the *LITTLE* cluster, as the reduction in power draw outweighs the increase in execution time compared to the *big* cluster.

Figure 6 illustrates the runtime of the benchmarks at all measured frequencies. We again chose only representative benchmarks to keep the figure legible, but the data for all benchmarks is available on our site. Dotted lines indicate running at the *big* cluster while solid lines indicate the *LITTLE* cluster. Again the difference between SP.A, which benefits tremendously from running at the *big* cluster is visible, compared to UA.A whose runtime benefits only in a minor fashion from running on the *big* cluster. In combination with Figure 5 this gives the complete picture, where we can trade faster execution against higher energy consumption for benchmarks that cannot significantly benefit from the features of the out-of order

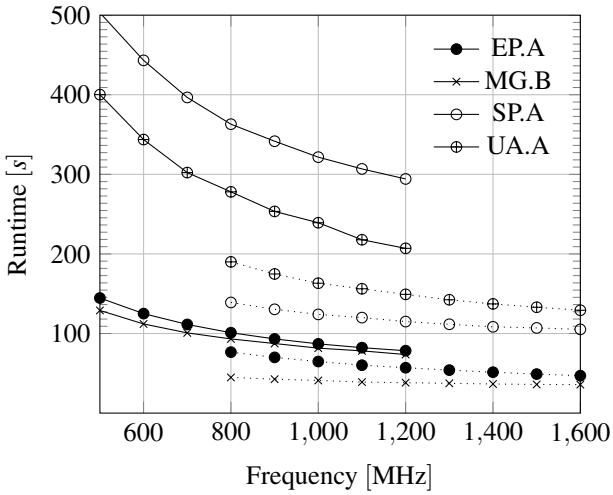

Figure 6: Runtime of NPB, solid lines denote *LITTLE* core, dashed lines denote *big* core

core.

The embarrassingly parallel (EP.A) benchmark, which has no dependencies between the threads and just computes random variates with compute load distributed evenly across the cores, is also interesting. It can be seen that this benchmark benefits very little from the architecture of the *big* core and thus gets only a marginal speedup when running on the A15 cluster at the same frequency as on the A7 cluster.

6 Effects of cluster switching

We also investigated the cost of migrating programs between the clusters. To achieve this we used two benchmarks from the NPB suite, one with high memory power draw (SP.A), one with very low memory power draw (DC.B). We executed the benchmark at 500 MHz on the

Type	L:b	E [J]	E' [J])	t [s]	E_{sw} [J]
M	1:0	1122.15 (5.21)	207.42	502.6	-
M	0:1	875.848 (1.81)	701.31	95.9	-
P	1:1	915.31 (-)	622.17	161.07	-
M	1:1	930.822 (3.54)	636.53	161.7	0.096
M	4:4	935.633 (2.83)	632.85	166.4	0.123
P	1:3	890.58 (-)	671.77	120.22	-
M	1:3	887.63 (2.76)	670.61	119.24	?
P	3:1	965.51 (-)	521.51	243.95	-
M	3:1	994.926 (3.03)	542.49	248.6	0.237

Table 1: Switching cost for SP.A (memory heavy)

Type	L:b	E [J]	E' [J])	t [s]	E_{sw} [J]
M	1:0	536.99 (2.62)	19.93	284.1	-
M	0:1	309.82 (3.64)	175.29	74.9	-
P	1:1	357.21 (-)	141.46	118.54	-
M	1:1	344.49 (2.24)	131.04	117.3	?
M	4:4	358.86 (2.63)	140.82	119.8	0.28
P	1:3	328.17 (-)	161.1	91.8	-
M	1:3	326.2 (2.54)	159.34	91.7	?
P	3:1	410.14 (-)	105.68	167.28	-
M	3:1	404.19 (2.27)	102	166.0	?

Table 2: Switching cost for DC.B (light on memory)

LITTLE cluster and at 1.6 GHz on the *big* cluster using 4 threads. We then performed the same benchmark and switched between the two frequencies in various intervals. We expected the runtime and the energy consumption of the switching execution to be the combination of the values for the individual executions distrbuted according to core ratio plus any overhead. The determination of this overhead was the original reason of the experiment. The results for running the benchmarks can be seen in Table 1 and Table 2.

The first columns indicate whether the values are measured (M) or predicted (P). Then follows the ratio between the cores. 1:1 indicates switching between the clusters every second, 1:3 means having the *LITTLE* cluster enabled for one second and then the *big* cluster for 3 seconds. *LITTLE* means running at 500 MHz on the A7 and *big* means running at 1.6 GHz on the A15 cluster. Predicted values assume that the energy consumption and runtime are the combination of the individual cluster values (row one and two) weighted according to core ratio.

The expected execution time is calculated according to the formula:

$$c = \frac{t_{little} \cdot t_{big}}{r_{little} \cdot t_{big} + r_{big} \cdot t_{little}}$$

where r_x denotes the ratio of core x as presented in the table, t_x is the execution time on core x and c is the number of switching cycles. One switching cycle contains the time on both cores and the switch between them. Energy consumptions were calculated accordingly.

The E' column is only the dynamic energy consumption for the benchmark while E includes static energy. The numbers in the E_{sw} column indicate the overhead of switching (predicted E - measured E divided by the number of switches) or a ? if it was negative due to misprediction.

As can be seen from the numbers switching clusters is more costly in terms of energy when looking at workloads with heavy memory usage. For light memory usage the presumed cost is way below the standard deviation and thus negligible. The predictions of the energy consumption and timing behavior using expected core ratios are reliable enough though we expect stronger mispredictions if applications with strong phasing behavior are used.

7 Conclusion and Future Work

In this paper we evaluated a heterogeneous architecture based on ARM *big.LITTLE*. We looked at the trade-offs for network, storage and applications and found that we can trade energy against latency and (on wireless) bandwidth, peak power against I/O speed and energy against execution time respectively. Especially the latter presents many trade-offs because there is no universal 'best core' or 'best frequency'. Some applications will perform best on the smallest frequency of the *LITTLE* core others will do so at the largest frequency. The same goes for the big core. Also the trade-offs and the gains to be found when executing on the *big* cluster vary heavily dependent on the application. Some may gain nothing at all (except more energy consumption) while others improve vastly. Also predicting the overhead of migrating between the clusters is non-trivial when it comes to time and, again, depends on the application characteristics when it comes to (dynamic) energy consumption.

These trade-offs present a challenge for future, energy-aware systems which we plan to address with our Energy/Utility [7] concept. We will implement this concept on the platform presented in the paper and show the benefit of a holistic, generic resource management framework that can cope with such highly heterogeneous platforms as presented here. We will also extend our work to the more recent Exynos5 Octa 5422 which enables the operation of both clusters in parallel.

Acknowledgments This work was partially funded by the German Research Council (DFG) through the Collaborative Research Center CRC 912 "Highly-Adaptive Energy- Efficient Systems" (HAEC) and the cluster of excellence "Center for Advancing Electronics Dresden".

References

[1] Exynos 5 octra second generation uses eight cores simultaneously. http://newsrls.com/news/exynos-5-octa-second-generation-use-eight-cores-simultaneously, Sep 2013.

[2] AKSANLI, B., ROSING, T., AND PETTIS, E. Distributed battery control for peak power shaving in datacenters. In *Green Computing Conference (IGCC), 2013 International* (June 2013), pp. 1–8.

[3] BAILEY, D. H. Nas parallel benchmarks. In *Encyclopedia of Parallel Computing*, D. A. Padua, Ed. Springer, 2011, pp. 1254–1259.

[4] ESMAEILZADEH, H., BLEM, E., ST. AMANT, R., SANKARALINGAM, K., AND BURGER, D. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture* (New York, NY, USA, 2011), ISCA '11, ACM, pp. 365–376.

[5] GUPTA, V., BRETT, P., KOUFATY, D., REDDY, D., HAHN, S., SCHWAN, K., AND SRINIVASA, G. The forgotten 'uncore': On the energy-efficiency of heterogeneous cores. In *Proceedings of the 2012 USENIX Conference on Annual Technical Conference* (Berkeley, CA, USA, 2012), USENIX ATC'12, USENIX Association, pp. 34–34.

[6] HÄHNEL, M., DÖBEL, B., VÖLP, M., AND HÄRTIG, H. ebond: Energy saving in heterogeneous r.a.i.n. In *Proc. of the Fourth International Conference on Future Energy Systems* (2013), e-Energy '13, ACM, pp. 193–202.

[7] HARTIG, H., VOLP, M., AND HAHNEL, M. The case for practical multi-resource and multi-level scheduling based on energy/utility. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2013 IEEE 19th International Conference on* (Aug 2013), pp. 175–182.

[8] HRUBY, T., BOS, H., AND TANENBAUM, A. S. When slower is faster: On heterogeneous multicores for reliable systems. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference* (Berkeley, CA, USA, 2013), USENIX ATC'13, USENIX Association, pp. 255–266.

[9] OMAR, N., BOSSCHE, P. V. D., COOSEMANS, T., AND MIERLO, J. V. Peukert revisitedcritical appraisal and need for modification for lithium-ion batteries. *Energies 6*, 11 (2013), 5625–5641.

[10] PATHAK, A., HU, Y. C., AND ZHANG, M. Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM European Conference on Computer Systems* (New York, NY, USA, 2012), EuroSys '12, ACM, pp. 29–42.

Notes

[1] http://odroid.com/dokuwiki/doku.php?id=en:odroid-xu
[2] http://hardkernel.com
[3] http://www.ti.com/product/ina231
[4] www.microchip.com/LAN9730
[5] https://github.com/TUD-OS/benchcat
[6] https://dbench.samba.org/
[7] https://github.com/TUD-OS