Green Computing – Prof. Dr. Bettina Schnor

Final project

Thema 6 Power Capping durch RAPL

Hanna Kretz

Prozess

Installation naaice1 und EMA	bis 02.08.
1. Einarbeitung	1 Woche bis 7.08.
2. Entwurf des Experimentes	2 Wochen bis 21.08.
3. Durchführung	1 Woche bis 28.08.
4. Auswertung der Ergebnisse	1 Woche bis 04.09.
5. Dokumentation	

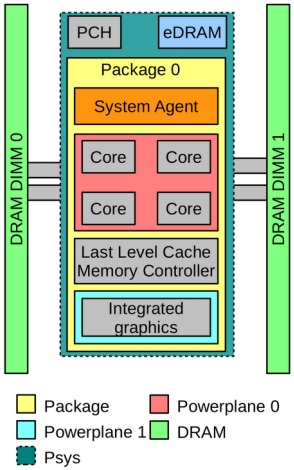
1. Einarbeitung

- Wiederholung C
- Literaturrecherche
  - Spazier, J. (2024). Präsentation “09-EMA”. *Green Computing.*
  - Hähnel, M., Döbel, B., Völp, M., & Härtig, H. (2012). Measuring energy consumption for short code paths using RAPL. *ACM SIGMETRICS Performance Evaluation Review*, *40*(3), 13-17.
  - Running Average Power Limit Energy Reporting, CVE-2020-8694 , CVE-2020-8695 / INTEL-SA-00389
  - Khan, K. N., Hirki, M., Niemi, T., Nurminen, J. K., & Ou, Z. (2018). Rapl in action: Experiences in using rapl for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, *3*(2), 1-26.

1. Einarbeitung

RAPL – Running Average Power Limit Energy Reporting

- Programmgesteuerte Messung des Energieverbrauchs von Intel Prozessoren
- Verfügbar seit 2011
- Liefert akkumulierte Verbrauchswerte in Mikrojoule (µJ)
- 1000 Samples pro Sekunde (1 ms)
  - Gründe der Sicherheit
- Keine pro-Core-Messungen
- Plugin bei EMA



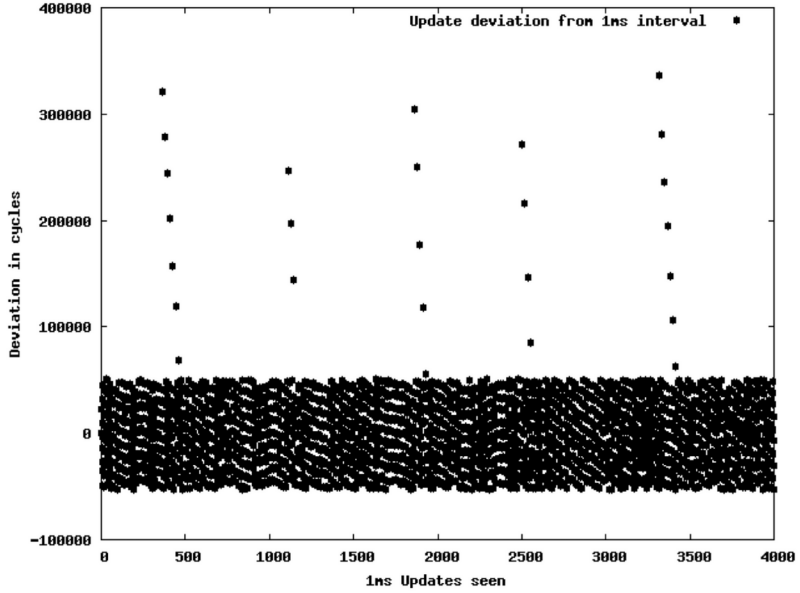
Quelle: Spazier, J. (2024). Präsentation “09-EMA”. *Green Computing.*

1. Einarbeitung

Hähnel et al. (2012):
Measuring energy consumption for short code paths using RAPL

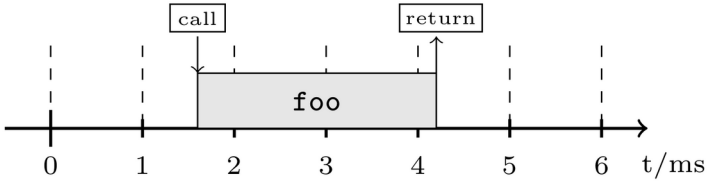
- Ist RAPL akkurat genug um Kurzzeit Messungen durchzuführen?
  - Nein, die Messfrequenz ist zu gering für genaue Messungen
- Implementierung von HEACER
  - Framework für Kurzzeit-Energie-Messungen

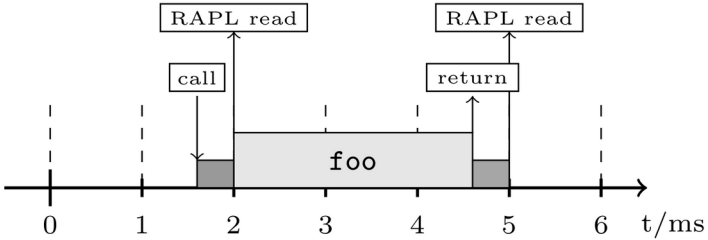
1. Einarbeitung



Quelle: Hähnel, M., Döbel, B., Völp, M., & Härtig, H. (2012). Measuring energy consumption for short code paths using RAPL. *ACM SIGMETRICS Performance Evaluation Review*, *40*(3), 13-17.

1. Einarbeitung





Quelle: Hähnel, M., Döbel, B., Völp, M., & Härtig, H. (2012). Measuring energy consumption for short code paths using RAPL. *ACM SIGMETRICS Performance Evaluation Review*, *40*(3), 13-17.

Forschungsfrage

Wie kurz kann eine Belastung auf die CPU dauern ($\leq$ 1 ms), dass RAPL es erkennt und aufzeichnet?

RAPL – Running Average Power Limit Energy Reporting

2. Entwurf des Experimentes

- Definition des Zieles: Bestimmen der „Reaktionszeit“
- Versuchsaufbau
  - Implementierung von Lastszenarien
  - CPU-intensive Berechnungen, die plötzlich starten und stoppen (Random-Number-Generator)
  - kontrollierte Intervalle (Dauer variieren, sleep-times dazwischen)
  - 11 Intervalllängen (0 ms, 0.1 ms, 0.2 ms, …, 0.9 ms, 1 ms)
  - 10.000 Messungen pro Intervall-Länge

2. Entwurf des Experimentes

- Messmethode
  - EMA zur Überwachung und Aufzeichnung des Energieverbrauchs
  - Erfassung der Energieverbrauchsdaten in kurzen Intervallen (jede Millisekunde)
  - Datenspeicherung: CSV

2. Entwurf des Experimentes – Quellcode

Header und Funktionsprototypen

```
1   #include <EMA.h>
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <unistd.h> // For usleep or sleep functions
5   #include <sys/time.h> // For time measurement
6
7   long long get_current_time_in_microseconds();
8   void do_work(int dur);
```

2. Entwurf des Experimentes – Quellcode

Funktionen

```
150  // FUNKTIONS
151
152  long long get_current_time_in_microseconds() {
153      struct timeval tv;
154      gettimeofday(&tv, NULL);
155      return (int)(tv.tv_sec * 1000000 + tv.tv_usec);
156  }
157
158  void do_work(int dur){
159      srand(1024);
160      long long t_before = get_current_time_in_microseconds();
161      long long t_new;
162      do{
163          rand();
164          t_new = get_current_time_in_microseconds();
165      } while(t_new - t_before < dur);
166  }
```

Aufrufdauer liegt bei ~ 40 ns

2. Entwurf des Experimentes – Quellcode

Main - Initialisierung

```
12  int main(int argc, char **argv)
13  {
14      int err = EMA_init(NULL); // Initialize EMA
15
16      if( err )       // Check for errors. If `err` is 0 then no error occurred.
17          return 1;
18
19      Filter *filter = EMA_filter_exclude_plugin("NVML"); // Create a filter to disable the NVML plugin.
20
21      // Initialize a region.
22      EMA_REGION_DECLARE(region_v0); // declares a region handle that holds the measurement data
23      EMA_REGION_DECLARE(region_v1);
24      EMA_REGION_DECLARE(region_v2);
25      EMA_REGION_DECLARE(region_v3);
26      EMA_REGION_DECLARE(region_v4);
27      EMA_REGION_DECLARE(region_v5);
28      EMA_REGION_DECLARE(region_v6);
29      EMA_REGION_DECLARE(region_v7);
30      EMA_REGION_DECLARE(region_v8);
31      EMA_REGION_DECLARE(region_v9);
32      EMA_REGION_DECLARE(region_v10);
```

2. Entwurf des Experimentes – Quellcode

Main - Initialisierung

```
34      // EMA_REGION_DEFINE` defines the region by setting a name
35      EMA_REGION_DEFINE_WITH_FILTER(&region_v0, "v0", filter); //is an extended version that takes an optional filter argument
36      EMA_REGION_DEFINE_WITH_FILTER(&region_v1, "v1", filter);
37      EMA_REGION_DEFINE_WITH_FILTER(&region_v2, "v2", filter);
38      EMA_REGION_DEFINE_WITH_FILTER(&region_v3, "v3", filter);
39      EMA_REGION_DEFINE_WITH_FILTER(&region_v4, "v4", filter);
40      EMA_REGION_DEFINE_WITH_FILTER(&region_v5, "v5", filter);
41      EMA_REGION_DEFINE_WITH_FILTER(&region_v6, "v6", filter);
42      EMA_REGION_DEFINE_WITH_FILTER(&region_v7, "v7", filter);
43      EMA_REGION_DEFINE_WITH_FILTER(&region_v8, "v8", filter);
44      EMA_REGION_DEFINE_WITH_FILTER(&region_v9, "v9", filter);
45      EMA_REGION_DEFINE_WITH_FILTER(&region_v10, "v10", filter);
46      // Alternatively: `EMA_REGION_DEFINE(&region, "region");`
47
48
49      int block_size = 1000; // 1ms
50      int replication_num = 10000; // block size * replication num = 10 s
```

2. Entwurf des Experimentes – Quellcode

Main - Messung

```
54      int duration = 0;
55      EMA_REGION_BEGIN(region_v0); //starts a measurement and stores the data in the region handle.
56      for(int x=0; x<replication_num; x++){
57          do_work(duration);
58          usleep(block_size-duration);
59      }
60      EMA_REGION_END(region_v0); //stops a measurement and stores the data in the region handle.
61
62      duration = 100;
63      EMA_REGION_BEGIN(region_v1);
64      for(int x=0; x<replication_num; x++){
65          do_work(duration);
66          usleep(block_size-duration);
67      }
68      EMA_REGION_END(region_v1);
```

2. Entwurf des Experimentes – Quellcode

Main – Messung und Finish

```
126     duration = 900;
127     EMA_REGION_BEGIN(region_v9);
128     for(int x=0; x<replication_num; x++){
129         do_work(duration);
130         usleep(block_size-duration);
131     }
132     EMA_REGION_END(region_v9);
133
134     duration = 1000;
135     EMA_REGION_BEGIN(region_v10);
136     for(int x=0; x<replication_num; x++){
137         do_work(duration);
138         usleep(block_size-duration);
139     }
140     EMA_REGION_END(region_v10);
141
142
143     EMA_filter_finalize(filter); // releases a previously created filter. This is just required if a filter was initialized.
144
145     EMA_finalize(); // Finalize EMA. This cleans up the EMA framework and prints out the measurement data.
146
147     return 0;
148 }
149
```

Erwartung

- Bei sehr kurzen CPU-Belastungen wird kein Energieverbrauch gemessen
- Wenn doch, ist es Zufall, dass die RAPL-Messung zu diesem Moment war
- Je länger die CPU-Belastungen desto wahrscheinlicher wird eine Energieverbrauchsmessung → höherer Energieverbrauch
- Gemessene Zeit pro Messung sollten 10 s entsprechen

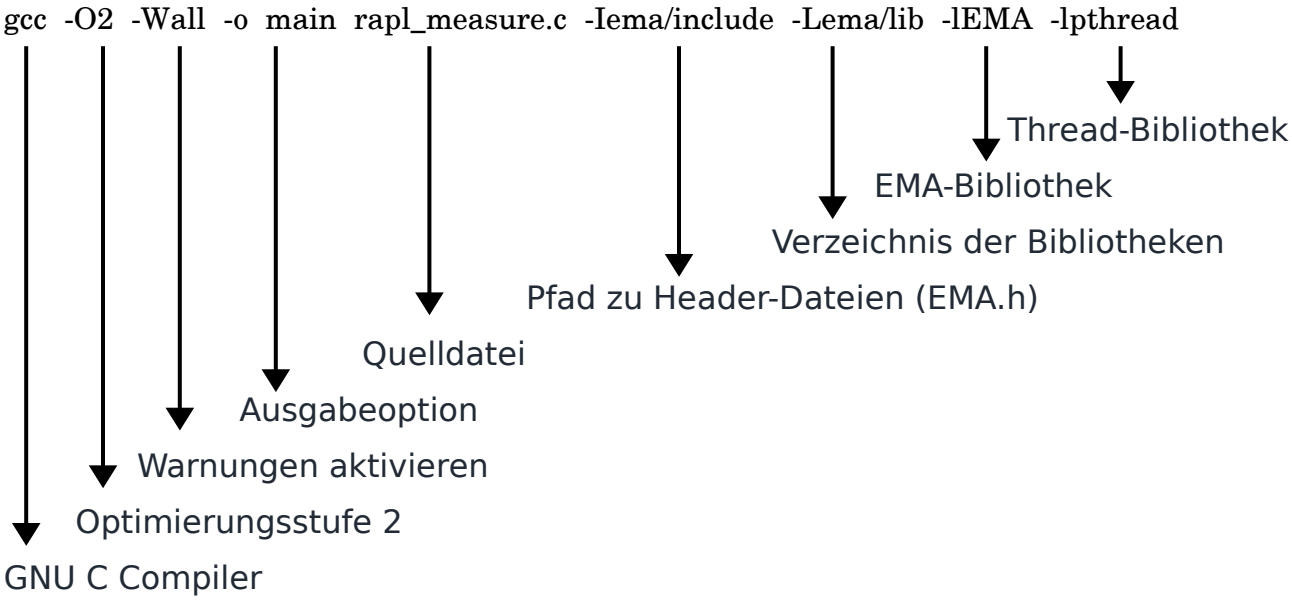
3. Durchführung

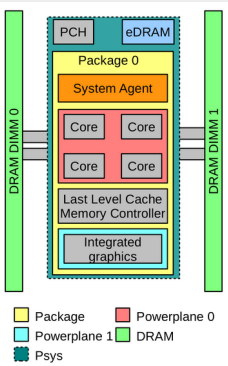
- Testlauf auf meinem Rechner
  - ThinkPad-T450
  - Intel® Core™ i5-5300U CPU @ 2.30GHz × 4
- Dann Durchführung auf naaice1
  - Dell Precision 3660 Tower
  - CPU: 1x Intel Core i9-12900 CPU @ 2.40GHz, 8C+8c/24T
  - Memory: 128GiB DIMM DDR5 Synchronous Registered (Buffered) 3600 MHz (0,3 ns)
  - Network
    - NetXtreme BCM5720 2-port Gigabit Ethernet
    - NVIDIA MCX623106AN-CDAT 2-port [ConnectX-6 Dx]

Quelle: Max Lübke

3. Durchführung

Kompilierung





3. Durchführung

Rohdaten

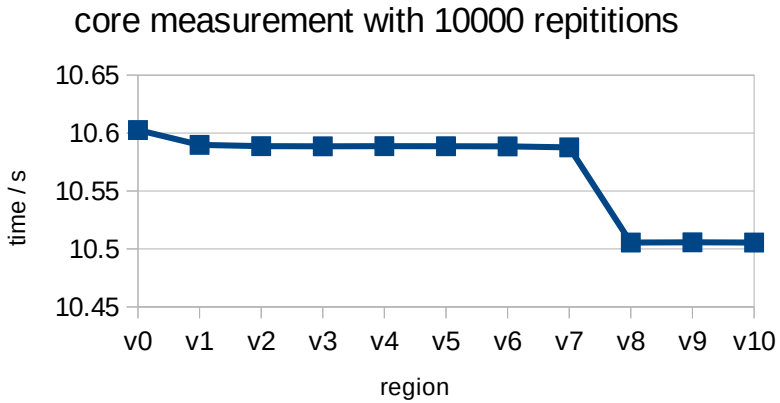
	A	B	C	D	E	F	G	H	I
1	thread	region_idf	file	line	function	visits	device_name	energy	time
2	0	v0	rapl_measure.c	51	main	1	CPU-0.package-0	122131889	10602552
3	0	v0	rapl_measure.c	51	main	1	CPU-0.core	17888627	10602605
4	0	v0	rapl_measure.c	51	main	1	CPU-0.uncore	1892	10602687
5	0	v1	rapl_measure.c	52	main	1	CPU-0.package-0	121547480	10589681
6	0	v1	rapl_measure.c	52	main	1	CPU-0.core	17634598	10589796
7	0	v1	rapl_measure.c	52	main	1	CPU-0.uncore	305	10589790
8	0	v2	rapl_measure.c	53	main	1	CPU-0.package-0	121352778	10588615
9	0	v2	rapl_measure.c	53	main	1	CPU-0.core	17307389	10588690
10	0	v2	rapl_measure.c	53	main	1	CPU-0.uncore	1343	10588688
11	0	v3	rapl_measure.c	54	main	1	CPU-0.package-0	121606073	10588523
12	0	v3	rapl_measure.c	54	main	1	CPU-0.core	17665605	10588595
13	0	v3	rapl_measure.c	54	main	1	CPU-0.uncore	1831	10588595
14	0	v4	rapl_measure.c	55	main	1	CPU-0.package-0	121645196	10588619
15	0	v4	rapl_measure.c	55	main	1	CPU-0.core	17548356	10588690
16	0	v4	rapl_measure.c	55	main	1	CPU-0.uncore	1709	10588703
17	0	v5	rapl_measure.c	56	main	1	CPU-0.package-0	121929498	10588609

4. Auswertung der Ergebnisse

Dauer der Messungen (11 Stück)

- Mittelwert t = 10,5672 s
- Standardabweichung s = 0,039 s

```
EMA_REGION_BEGIN(region_v1);
for(int x=0; x<replication_num; x++){
    do_work(duration);
    usleep(block_size-duration);
}
EMA_REGION_END(region_v1);
```



4. Auswertung der Ergebnisse

Dauer der Messungen (11 Stück)

- Mittelwert t = 10,5672 s
- Standardabweichung s = 0,039 s

```
EMA_REGION_BEGIN(region_v1);
for(int x=0; x<replication_num; x++){
    do_work(duration);
    usleep(block_size-duration);
}
EMA_REGION_END(region_v1);
```

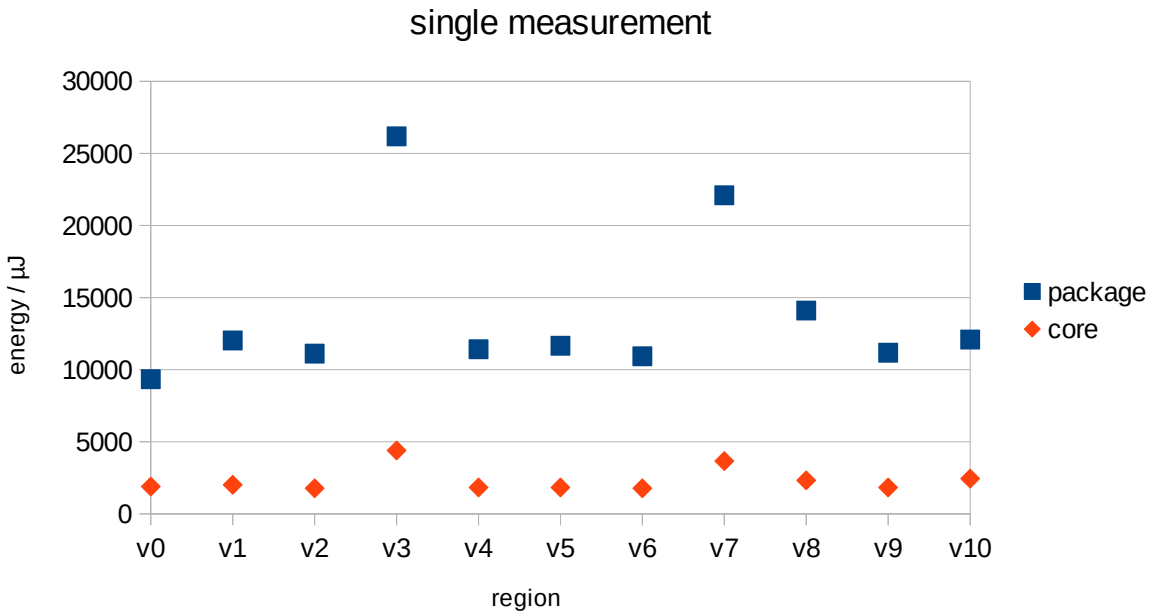
Dauer eines Schleifendurchlaufs (10.000 Stück)

- Mittelwert t = 1056 µs
- Standardabweichung s = 3,98 µs

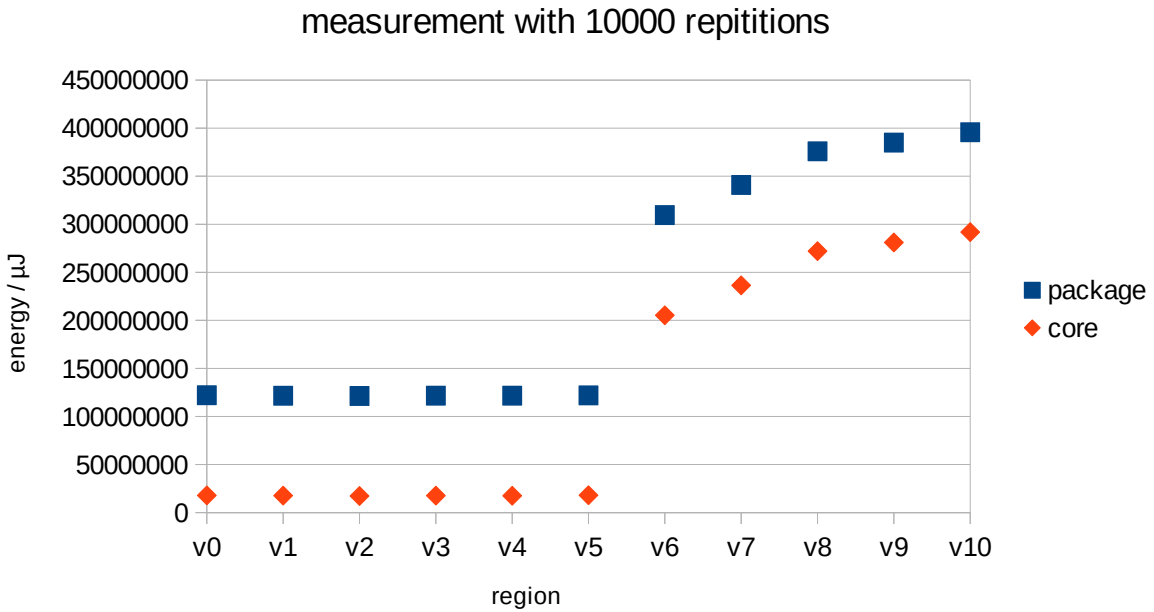
➔ Regelmäßige Verzögerung um ~52µs

➔ Ursache bspw. Funktionsaufruf o. Ä.

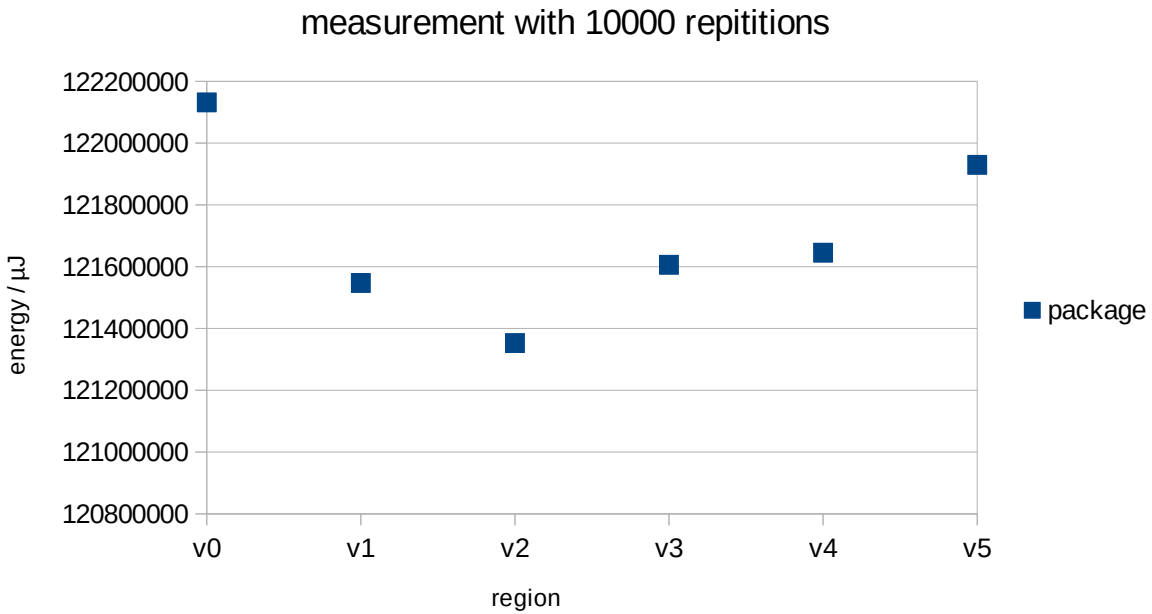
4. Auswertung der Ergebnisse



4. Auswertung der Ergebnisse



4. Auswertung der Ergebnisse



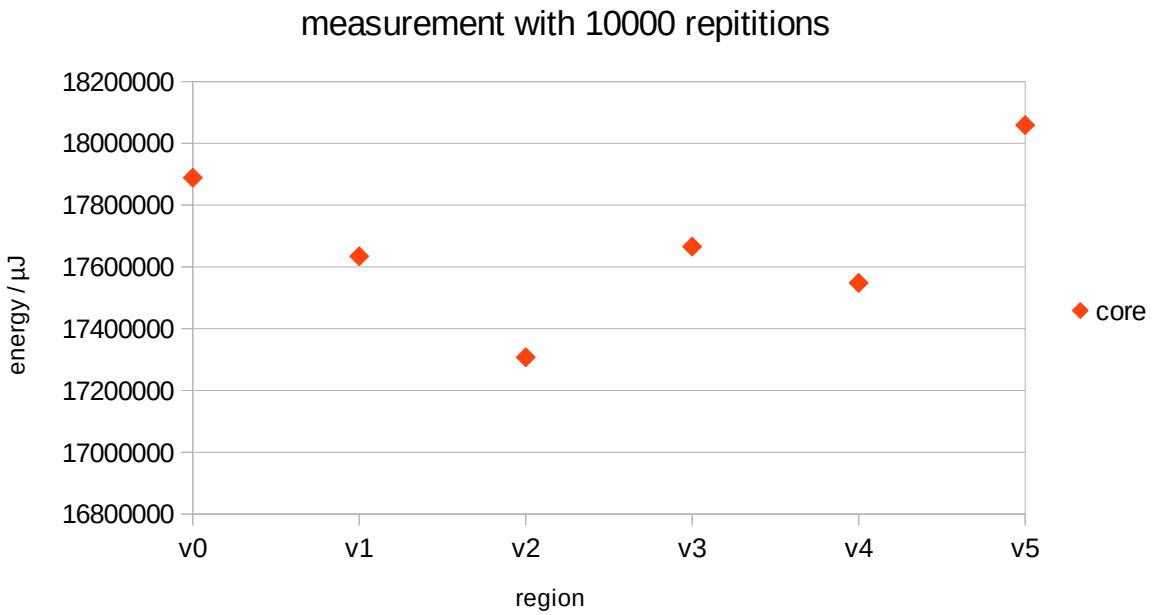
4. Auswertung der Ergebnisse

Energieverbrauch Package-0

- Min. E = 121,352 J
- Max. E = 395,768 J
- Mittelwert E = 230,653 J
- Standardabweichung s = 127,179 J

- Bei einer Belastungslänge von t = 0 ms bis t = 0,5 ms schwankt der Messwert um ø E = 121,702 J

4. Auswertung der Ergebnisse



4. Auswertung der Ergebnisse

Energieverbrauch Core

- Min. E = 17,307 J
- Max. E = 291,844 J
- Mittelwert E = 126,620 J
- Standardabweichung s = 127,187 J

- Bei einer Belastungslänge von t = 0 ms bis t = 0,5 ms schwankt der Messwert um ø E = 17,68 J

4. Auswertung der Ergebnisse

- ➔ Regelmäßige zeitliche Verzögerung jedes Messdurchlaufs um ~52µs
- ➔ Bei einer Belastungslänge von t = 0 ms liegt der gemessene Energieverbrauch nicht bei E = 0 µJ.
- ➔ Bei t = 0 ms bis t = 0,5 ms ist keine eindeutige Steigung zu erkennen, eher eine zufällige Streuung
- ➔ Ab t = 0,6 ms steigt der Energiemesswert signifikant, es ist eine eindeutige Steigung des Energiemesswertes erkennbar

4. Auswertung der Ergebnisse - Interpretation und Schlussfolgerungen

- Unter 0,5 ms liegt die Wahrscheinlichkeit auch unter 50%, dass RAPL etwas von der Last mitbekommt, bei 10000 Wiederholungen kommt es aber trotzdem hin und wieder vor, dass RAPL die Last “erwischt”
- Bei längeren Belastungen steigt die Wahrscheinlichkeit, dass RAPL einen Energieverbrauch misst

➔ Eine Energiemessung ist bei Kurzzeitmessungen (besonders unter t = 0,5 ms) dem Zufall überlassen und bringt keine belastbaren Messergebnisse

4. Auswertung der Ergebnisse - Beantwortung der Forschungsfrage

Wie kurz kann eine Belastung auf die CPU dauern (≤ 1 ms), dass RAPL es erkennt und aufzeichnet?

➔ Ab einer Belastungsdauer von t = 0,6 ms scheint RAPL recht verlässlich einen Energieverbrauch zu messen.

4. Fehleranalyse und Reflexion

Messung

- Messwerte sind durch hohe Wiederholrate belastbar, Störvariablen können jedoch nicht ausgeschlossen werden
- In Einzelmessungen sind RAPL-Werte bei Kurzzeitmessungen nicht belastbar

Projektarbeit

- Späte Festlegung der Forschungsfrage
- Probleme bei Arbeit mit externem PC (naaice1) durch wenig Erfahrung
- Nicht an Zeitplan gehalten

Literatur

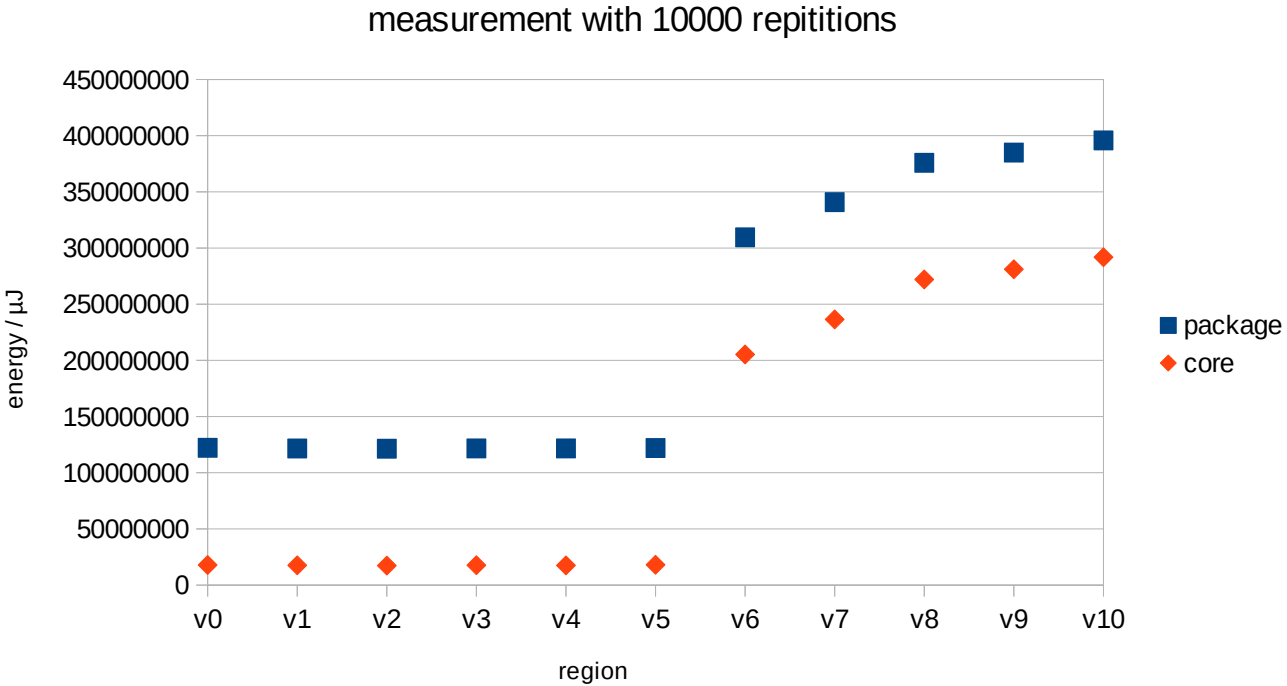
Hähnel, M., Döbel, B., Völp, M., & Härtig, H. (2012). Measuring energy consumption for short code paths using RAPL. *ACM SIGMETRICS Performance Evaluation Review*, *40*(3), 13-17.

Khan, K. N., Hirki, M., Niemi, T., Nurminen, J. K., & Ou, Z. (2018). Rapl in action: Experiences in using rapl for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, *3*(2), 1-26.

Running Average Power Limit Energy Reporting, CVE-2020-8694 , CVE-2020-8695 / INTEL-SA-00389

Spazier, J. (2024). Präsentation “09-EMA”. *Green Computing.*

Steinert, F. & Stabernack, B. (2023). FPGA-Based Network-Attached Accelerators – An Environmental Life Cycle Perspective. *ARCS-FPGA-KPI4DCE.*


measurement with 10000 repititions
energy / µJ
450000000
400000000
350000000
300000000
250000000
200000000
150000000
100000000
50000000
0
v0
v1
v2
v3
v4
v5
v6
v7
v8
v9
v10
region
package
core