HOCHSCHULE ESSLINGEN

Forschungsprojekt Bericht 1

Libiotrace CPU Power Measurement in HPC-Cluster

B. Eng. Andreas Heinrich

Studiengang: Angewandte Informatik
Fakultät: Informationstechnik
Hochschule Esslingen
Sommersemester 2023 und Wintersemester 2023/24

Zeitraum: 01.03.2023 - 23.02.2024
Dozent: Prof. Dr.-Ing. Rainer Keller
Betreuer: Philipp Köster

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Esslingen, den 23. Februar 2024 _____

Unterschrift

Abstract

Der vorliegende Bericht informiert über den aktuellen Stand des Forschungsprojektes „Libiotrace“, das sich mit der Entwicklung eines Tools beschäftigt, welches Live-Tracing-Daten von High-Performance-Computing (HPC) Anwendungen sammelt und in Echtzeit analysiert. Bisher konzentrierte sich das bestehende Tool auf die Erfassung und Auswertung von I/O-Daten. Diese soll nun um die Funktionalität für die Erfassung von Live-Energiedaten der CPU zu erweiterung werden.

Der Bericht stellt zwei spezifische Schnittstellen vor, die Live-Energiedaten der CPU zur Verfügung stellt und erläutert, wie diese genutzt werden können. Anschließend wird beschrieben, wie das Forschungsprojekt „Libiotrace“ um diese neue Funktionalität erweitert wurde. Abschließend werden noch Kennzahlen zu den Messungen aufgezeigt, um die Leistungsfähigkeit und Effizienz der neuen Erweiterung darzustellen. Diese Erweiterung ermöglicht eine umfassendere Analyse der Performance und Energieeffizienz von HPC-Anwendungen, was für die Optimierung dieser Systeme von entscheidender Bedeutung ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Codeverzeichnis

1 Einleitung

Das High-Performance-Computing (HPC) beschäftigt sich mit dem Hochleistungsrechnen. Im Bereich des High-Performance-Computing geht es darum, dass Computer mit möglichst großer Leistung und möglichst vielen parallelen Prozessen operieren. Energiedaten der CPU können dabei als Merkmal der Effizienz dienen, geben dem Anwender direktes Feedback über den tatsächlichen Verbrauch einer Anwendung und zeigen über die Zeit die Last der CPU und der umliegenden Komponenten.

1.1 Motivation

Die Erfassung und Auswertung von Energiedaten der CPU ist in der heutigen Zeit, in der Energieeffizienz und Nachhaltigkeit zunehmend an Bedeutung gewinnen, von entscheidender Wichtigkeit. Die präzise Erfassung und Analyse dieser Daten versetzen Entwickler und Forscher in die Lage, Energieverbrauchsmuster von HPC-Anwendungen zu verstehen und zu interpretieren. Durch das Verständnis, wie und wann die Energieverbrauchsspitzen auftreten, können Software und Hardware gezielt optimiert werden, um die Energieeffizienz zu steigern und somit die Betriebskosten zu senken.

Eine Optimierung der Energieeffizienz durch datengestützte Entscheidungen reduziert nicht nur die Betriebskosten, sondern hat auch eine positive Auswirkung auf die Umwelt. Eine Auswertung von Energiedaten der CPU ist somit ein wichtiger Schritt hin zu intelligenteren, kosteneffizienteren und umweltfreundlicheren Computertechnologien.

1.2 Ziel

Ziel meiner Forschungsarbeit soll sein, Energiedaten der CPU durch das Erweitern der „Libiotrace“ für Entwickler aufzubereiten und somit ein einfaches und direktes Feedback einer HPC-Anwendung zu liefern.

Das Ziel wird in zwei Forschungsabschnitte aufgeteilt:

- **Abschnitt 1**:
  Das Ziel des ersten Abschnittes ist es, die unterschiedlichen Schnittstellen der CPU zu evaluieren und aufzuzeigen, sowie die Erweiterung der „Libiotrace“ um das erfassen von Live-Energiedaten.
- **Abschnitt 2**:
  Das Ziel des zweiten Abschnittes ist es, die gesammelten Live-Energiedaten für die Entwickler aufzubereiten, sowie das Messen von Kennzahlen und die Optimierung dieser.

1.3 Struktur

Der Bericht zeigt zu Beginn die Ergebnisse der Evaluation der Schnittstellen. Anschließend wird noch aufgezeigt, wie die „Libiotrace“ um die Messung der Live-Energiedaten erwei-

tert wurde. Zuletzt werden noch erste Messungen der Live-Energiedaten und die damit verbunden Möglichkeiten aufgezeigt.

2 Energiedaten-Messung

Um die CPU-Energiedaten einer HPC Anwendung zu messen, stehen verschiedene Methoden zur Verfügung. Im folgenden Kapitel werden diese unterschiedlichen Ansätze vorgestellt.

2.1 Hardware-Counter

Eine CPU besitzt verschiedene Hardware-Counter. Hardware-Counter sind Werkzeuge zum Verständnis von Softwareleistung und Energieverbrauch [4]. Diese Counter können verschiedene Ereignisse wie „data loads“, „cache misses“ und „retired instructions“ überwachen [5]. Dennoch ist anzumerken, dass in der Praxis durchgeführte Implementierungen von Performance Monitoring Units (PMUs) nicht zwingend exakte und deterministische Daten generieren. Dies resultiert aus den inhärenten Variationen während des Betriebs sowie aus der Neigung zu Überzählungen auf Maschinen mit x86_64-Architektur. [6]. Hardware-Counterbasierte Methoden zur Energie- und Leistungsmodellierung sowie Optimierung wurden erfolgreich angewandt, um den Energieverbrauch bei umfangreichen wissenschaftlichen Applikationen auf energieeffizienten Supercomputern zu minimieren. [7]. Trotz bestehender Einschränkungen sind Hardware-Counter wichtige Instrumente für die Leistungsanalyse und Energiemodellierung.

RAPL (Running Average Power Limit) und PowerCap sind Technologien, die von Intel entwickelt wurden, um den Energieverbrauch von Prozessoren pro Sockel zu steuern [8]. Mit dem Linux 5.8 sind die von Intel entwickelten Technologien auch für AMD Prozessoren verfügbar [9]. Dabei war es möglich Daten für jeden Core einer CPU auszulesen. Mit Linux Kernel Update 5.13 wurde dieses aufgrund der CVE-2020-12912 eingeschränkt [10, 11]. Dabei bieten beide Technologien die Möglichkeit, wie in Abb. 2.2 zusehen ist, Daten für unterschiedliche Leistungsbereiche auszulesen [1].

2.1.1 Running Average Power Limit

RAPL ermöglicht eine präzise Echtzeitüberwachung und Steuerung des Energieverbrauchs auf Mikroarchitekturebene. Hierfür werden die Daten aus `/dev/cpu/*/msr` gelesen. Die Messung erfolgt hierbei in Nanojoules [12].

Die Datei `/dev/cpu/*/msr` ist eine Systemdatei, die Zugriff auf die modellspezifischen Register (MSRs) der CPU ermöglicht. MSRs sind Steuerregister in der CPU, die privilegierter Software den Zugriff auf verschiedene Prozessoreinstellungen und Leistungsmerkmale ermöglichen [13]. Der Zugriff auf `/dev/cpu/*/msr` kann zu potenziellen Sicherheitsimplikationen führen, da er direkten Zugriff auf modellspezifische Register (MSRs) auf der CPU ermöglicht [14].

Auf Grundlage der Sicherheitsimplikationen ist der Zugriff nur auf diese Datei ausschließlich für Root Benutzer möglich und kann nicht über Berechtigung mit `chmod` erteilt werden.

RAPL wurde bereits ausgiebig für die Messung und Modellierung des Stromverbrauchs verwendet und gilt als genau genug, um den Energieverbrauch von Servern zu überwachen, ohne dass komplexe Stromzähler erforderlich sind [1].

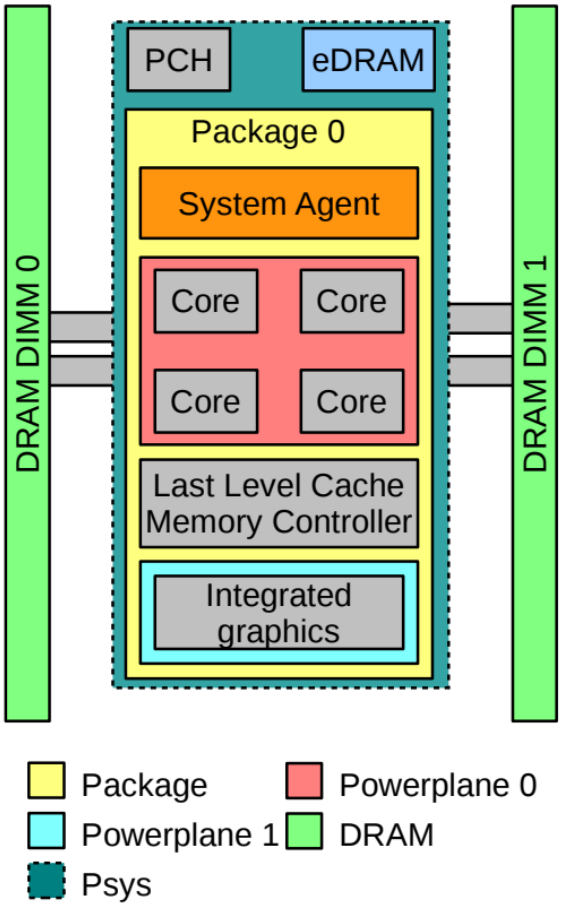

Abbildung 2.1: Übersicht der unterstützten Leistungsbereiche pro Sockel[1]

2.1.2 PowerCap

PowerCap ist spezifischer auf die Kontrolle des Energieverbrauchs ausgerichtet. Es ermöglicht nicht nur das Auslesen des Energieverbrauchs, sondern auch das Festlegen von Energiegrenzen der verschiedene Komponenten, um sicherzustellen, dass der Energieverbrauch bestimmte Schwellenwerte nicht überschreitet. Hierfür werden die Daten aus `/sys/class/powercap/intel-rapl:*/energy_uj` und `/sys/class/powercap/intel-rapl:*:*/energy_uj` verwendet. Die Messung erfolgt hierbei in Microjoules [15].

Eine Anwendung von PowerCap liegt in Supercomputern wie dem SuperMUC-NG. Der SuperMUC-NG ist ein Supercomputer im Leibniz-Rechenzentrum in Garching bei München. Um sicherzustellen, dass das System im Laufe der Zeit innerhalb eines Strombudgets bleibt, wird PowerCap eingesetzt. Dabei ermöglicht es kurzfristige Überschreitungen des Leistungslimits, solange der durchschnittliche Stromverbrauch unter dem festgelegten Grenzwert bleibt [16].

2.2 Messgerät am Netzteil

Eine weitere Methode, um die Energie von HPC-Anwendungen zu messen, ist das Verwenden von externen Messgeräten. Diese messen den Stromverbrauch des Netzteils und nicht direkt den Verbrauch der CPU. Der Verbrauch vom Netzteil beinhaltet den Verbrauch von allen verbauten Komponenten, wie CPU, RAM, Festplatten, PCI-Karten und Lüfter.

2.2.1 Shelly Plus Plug S

Der Shelly Plus Plug S ist ein Smart-WLAN-Stecker mit Bluetooth. Der Shelly Plug S dient als Referenzmessung zu dem Hardware-Counter, über die vorhandene REST-API können Live-Messungen der Verbrauchsdaten abgerufen werden [2]. Die API liefert verschiedene Werte, siehe Code 2.1. Für den Vergleich betrachten wir den aktuellen Verbrauch in Watt („apower“).

Abbildung 2.2: Shelly Plus Plug S [2]

```
$ curl http://192.168.178.96/rpc/Switch.GetStatus?id=0
{
    "id": 0,
    "source": "init",
    "output": true,
    "apower": 120.2,
    "voltage": 233.6,
    "current": 0.555,
    "aenergy": {
        "total": 68866.594,
        "by_minute": [
            1799.654,
            2094.687,
            2251.947
        ],
        "minute_ts": 1708678725
    },
    "temperature": {
        "tC": 39.0,
        "tF": 102.1
    }
}
```

Listing 2.1: REST-Call und Response der Shelly Plus Plug S API [3]

3 Implementierung

Zur Durchführung der Messungen wurden RAPL und PowerCap in die „Libiotrace“ integriert. Diese Integration ermöglicht die Echtzeiterfassung von Leistungsdaten des Systems und deren Übertragung zusammen mit anderen Messwerten an eine InfluxDB, wo sie gespeichert werden. Die Integration in die Libiotrace schafft eine nahtlose Verbindung zwischen verschiedenen Messquellen und ermöglicht eine effiziente Datenerfassung und -speicherung für weiterführende wissenschaftliche Untersuchungen. Alle Messpunkte werden von der libiotrace jede Sekunde ausgelesen und verarbeitet.

Die Datenerhebung erfolgt innerhalb des Feedback-Threads der InfluxDB, welcher in regelmäßigen Abständen durchläuft. Eine verstärkte Interaktion mit der InfluxDB kann eine Beeinflussung der Messintervalle zur Folge haben.

3.1 MAKEFILE Optionen

Um die „Libiotrace“ beim Bauen individuell konfigurieren zu können, wurden folgende Build Parameter in die `CMakeLists.txt` hinzugefügt.

3.1.1 POWER_MEASUREMENT_INTERVAL

`Default: 1000000000` ns

Mithilfe von „POWER_MEASUREMENT_INTERVAL“ kann die minimale Zeit zwischen den Messungen und damit die Messhäufigkeit in Nanosekunden konfiguriert werden. Die tatsächliche Zeit der Intervalle hängt von der Kommunikationsdauer zur InfluxDB ab.

3.1.2 ENABLE_POWER_MEASUREMENT_POWERCAP

`Default: OFF`

Durch die Verwendung von „ENABLE_POWER_MEASUREMENT_POWERCAP“ kann die Energiemessung über die PowerCap Schnittstelle (Siehe Kapitel 2.1.2 auf Seite 4) hinzugefügt werden.

3.1.3 ENABLE_POWER_MEASUREMENT_RAPL

`Default: OFF`

Mithilfe von „ENABLE_POWER_MEASUREMENT_RAPL“ kann die Energiemessung über die RAPL Schnittstelle (Siehe Kapitel 2.1.1 auf Seite 3) hinzugefügt werden.

3.1.4 ENABLE_POWER_MEASUREMENT_RAPL_PER_CORE

`Default: OFF`

Durch die Verwendung von „ENABLE_POWER_MEASUREMENT_RAPL_PER_CORE“ kann die Energiemessung von RAPL bei aktiviertem „ENABLE_POWER_MEASUREMENT

_RAPL“ auf alle CPU Cores erweitert werden. Je nach CPU Modell werden pro CPU Sockel dieselben Energiedaten bereitgestellt.

3.2 InfluxDB und Struct

Mittels dem Structs „power_measurement_data“ (Listing 3.1 auf Seite 7) werden durch Macros alle Funktionen erstellt, die benötigt werden um Datensätze für das Senden an die InfluxDB vorzubereiten. Über die Funktion `write_power_measurement_data_into_influxdb` werden die Messpunkte über einen POST-Request an die InfluxDB übertragen, wo die Messdaten zentral persistiert werden.

```
1  LIBIOTRACE_STRUCT_START(power_measurement_data)
2    LIBIOTRACE_STRUCT_U_INT64_T(time)
3    LIBIOTRACE_STRUCT_PID_T(pid)
4    LIBIOTRACE_STRUCT_INT(cpu_package)
5    LIBIOTRACE_STRUCT_INT(cpu_id)
6    LIBIOTRACE_STRUCT_CSTRING_P(name, 100)
7    LIBIOTRACE_STRUCT_INT(type)
8    LIBIOTRACE_STRUCT_U_INT64_T(measurement_difference_to_last_value)
9    LIBIOTRACE_STRUCT_U_INT64_T(measurement_convert_value)
10   LIBIOTRACE_STRUCT_U_INT64_T(measurement_value)
11 LIBIOTRACE_STRUCT_END
```

Listing 3.1: Struct „power_measurement_data“

Das Struct „power_measurement_data“ ist wie folgt aufgebaut:

- **time**: Zeitpunkt der Messung
- **pid**: Prozess Id der Messung
- **cpu_package**: Nummer des CPU Package (Sockel)
- **cpu_id**: Eindeutige Id des CPU Kerns
- **name**: Name des Messpunktes (z.B. „PACKAGE_ENERGY“, „DRAM_ENERGY“, etc.)
- **type**: Typ der Messung (z.B. `5` für „PACKAGE_ENERGY_CNT“, `10` für „DRAM_ENERGY“, etc.)
- **measurement_difference_to_last_value**: Differenz zur letzten Messung (finaler Wert für PowerCap)
- **measurement_convert_value**: (Nur RAPL) Konvertieren der Messung je nach Type (finaler Wert für RAPL)
- **measurement_value**: Originaler Wert aus der Datei

3.3 Ablauf der Messungen

Die Messungen werden nach dem Task-Pattern durchgeführt, dabei durchläuft der Prozess drei Phasen.

3.3.1 Phase 1: Initialisierung

Funktion: `power_measurement_init()` in `event.c`

In der Initialisierungsphase werden Systeminformationen ausgelesen und verarbeitet. Die Systemdatei `/proc/cpuinfo` wird für die Ermittlung des CPU-Modells und dessen Familie genutzt. Die Anzahl der CPU-Sockets und die dazugehörige CPU_Ids wird über die `/sys/devices/system/cpu/kernel_max` und `/sys/devices/system/cpu/cpu<cpu_id>/topology/physical_package_id` ermittelt.

Zudem werden `CPUMeasurementTask`-Objekte erstellt, die in der zweiten Phase analysiert werden sollen.

Nach dieser Ermittlung werden die aktivierten Schnittstellen initialisiert.

Initialisierung RAPL

Im Rahmen der Initialisierungsphase von RAPL werden, abhängig vom CPU-Modell und der zugehörigen Familie, verschiedene Messpunkte aktiviert, die spezifisch für die betreffende CPU verfügbar sind. Für diese ausgewählten Messpunkte werden daraufhin `CPUMeasurementTask`-Instanzen erstellt, die in Phase 2 verarbeitet werden. Abschließend wird ein Überprüfungsprozess durchgeführt, um sicherzustellen, dass die erforderlichen Zugriffsrechte zum Lesen der benötigten Dateien vorhanden sind und um die Initialisierung der Messpunkte abzuschließen.

Initialisierung PowerCap

Im Rahmen der Initialisierungsphase von PowerCap werden die vorhanden Messpunkte unter `/sys/class/powercap/intel-rapl*/*` analysiert und zu `CPUMeasurementTask`-Instanzen umgewandelt, die in Phase 2 verarbeitet werden.

3.3.2 Phase 2: Messungen

Funktion: `power_measurement_step()` in `event.c`

Die Funktion wird zyklisch aufgerufen, solange die „Libiotrace“ läuft. Ist die Zeitdifferenz zum letzten Aufruf der Funktion größer oder gleich dem Minimumintervall `POWER_MEASUREMENT_INTERVAL` (siehe dazu Kapitel 3.1.1 auf Seite 6), so werden alle in Phase 1 angelegten `CPUMeasurementTask` ausgeführt, verarbeitet und die Ergebnisse mittels dem „`power_measurement_data`“-Struct aus Kapitel 3.2 auf Seite 7 an die InfluxDB gesendet.

3.3.3 Phase 3: Cleanup

Funktion: `power_measurement_cleanup()` in `event.c`

Beim Beenden der „Libiotrace“ werden alle allokierten Speicherbereiche wieder freigegeben.

3.4 Shelly Plug S

Mittels einem Python Skripts unter `libiotrace/scripts/shelly-to-influx` im Projekt werden die Daten der API aus Kapitel 2.2.1 auf Seite 5 jede Sekunde abgerufen und an

die InlfuxDB gesendet. Für die Verbindung zur InfluxDB nutzt das Skript dieselben Umgebungsvariablen (Environment Parameters), die auch von „Libiotrace“ verwendet werden. Diese Variablen werden aus einer `.env`-Datei ausgelesen.

Um einen direkten Vergleich der Messpunkte mit denen der Hardware-Counter zu ermöglichen, wird der Wert „apower“ in Watt für die Dauer von einer Sekunde verwendet. Dadurch ergibt sich folgende Berechnung:

$$1\ Watt\ [W]\ *\ 1\ Sekunde\ [s]\ =\ 1\ Wattsekunde\ [Ws]\ =\ 1\ Joule\ [J]$$

3.5 Herausforderungen

Aufgrund der erforderlichen Root-Rechte, die notwendig sind, damit ein Prozess Lesezugriff auf die benötigten Dateien erhält, müssen „Libiotrace“ und somit auch das zu messende Programm mit Root-Rechten ausgeführt werden. Dies kann je nach System und Programm zu Sicherheitsrisiken führen. Messungen sind deshalb auf dem bwUniCluster nicht möglich.

4 Messungen

Im Folgenden werden erste Messungen der Implementierung aufgezeigt. Als Referenzmessung dient der Stromverbrauch des gesamten Systems, der durch einen Shelly Plus Plug S, wie in Kapitel 3.4 auf Seite 8 erläutert, gemessen wird. Die Messungen wurden auf unterschiedlicher Hardware mit verschiedenen CPUs durchgeführt. Während der Messungen waren auf den Systemen weitere Prozesse parallel aktiv, es handelte sich somit nicht um unbelastete Systeme. Die verfügbaren Leistungsbereiche variieren je nach CPU-Familie und CPU-Modell.

4.1 Hardware Informationen

Folgende Hardware wurde für die Messungen verwendet. Die InfluxDB wurde auf einem bwCloud-Server unter Verwendung von Docker betrieben.

Hostname	Bezeichnung	Linux-Distribution	Kernel Version
itlx0028.hs-esslingen.de	Betriebssysteme Server	Rocky Linux release 9.3	5.14.0-362.8.1.el9_3.x86_64
worktower	Privater Heimrechner	Arch Linux	6.6.10-arch1-1

Tabelle 4.1: Host Informationen

Hostname	Model name	CPU family	CPU Model	Socket(s)	Hyper-Threading
itlx0028.hs-esslingen.de	Intel(R) Xeon(R) Gold 6150 CPU @ 2.70GHz	6	85	4	Nein
worktower	AMD Ryzen 9 3900X 12-Core Processor	23	113	1	Ja

Tabelle 4.2: CPU Informationen

Hostname	Package	Powerplane 0	Powerplane 1	DRAM	Psys	Shelly Plus Plug S
itlx0028.hs-esslingen.de	ja	nein	nein	ja	nein	nein
worktower	ja	ja	nein	nein	nein	ja

Tabelle 4.3: Mögliche Messungen

4.2 Test Programm 1

Um die verschiedenen Leistungsbereiche zu messen, wurde ein Demo-Programm geschrieben, das folgende Zyklen mehrmals hintereinander ausführt.

- Startet CPU Last
  - Erstellt `F` Forks
  - Iteriert in jedem Fork `K`-mal von 0 bis 1048576
  - Wartet bis alle Forks fertig und beendet sind
- 15 Sekunden Pause

- Startet RAM Last
  - Erstellt `F` Forks
  - Allokiert in jedem Fork `S * sizeof(int)`-mal Speicher
  - Iteriert in jedem Fork `I`-mal über den allokierten Speicher und schreibt und liest an der aktuellen Position
  - Gibt den allokierten Speicher wieder frei
  - Wartet bis alle Forks fertig und beendet sind
- 15 Sekunden Pause

Dieser Prozess wird in Abbildung 4.1 auf Seite 12 in Form eines Flussdiagramms veranschaulicht.

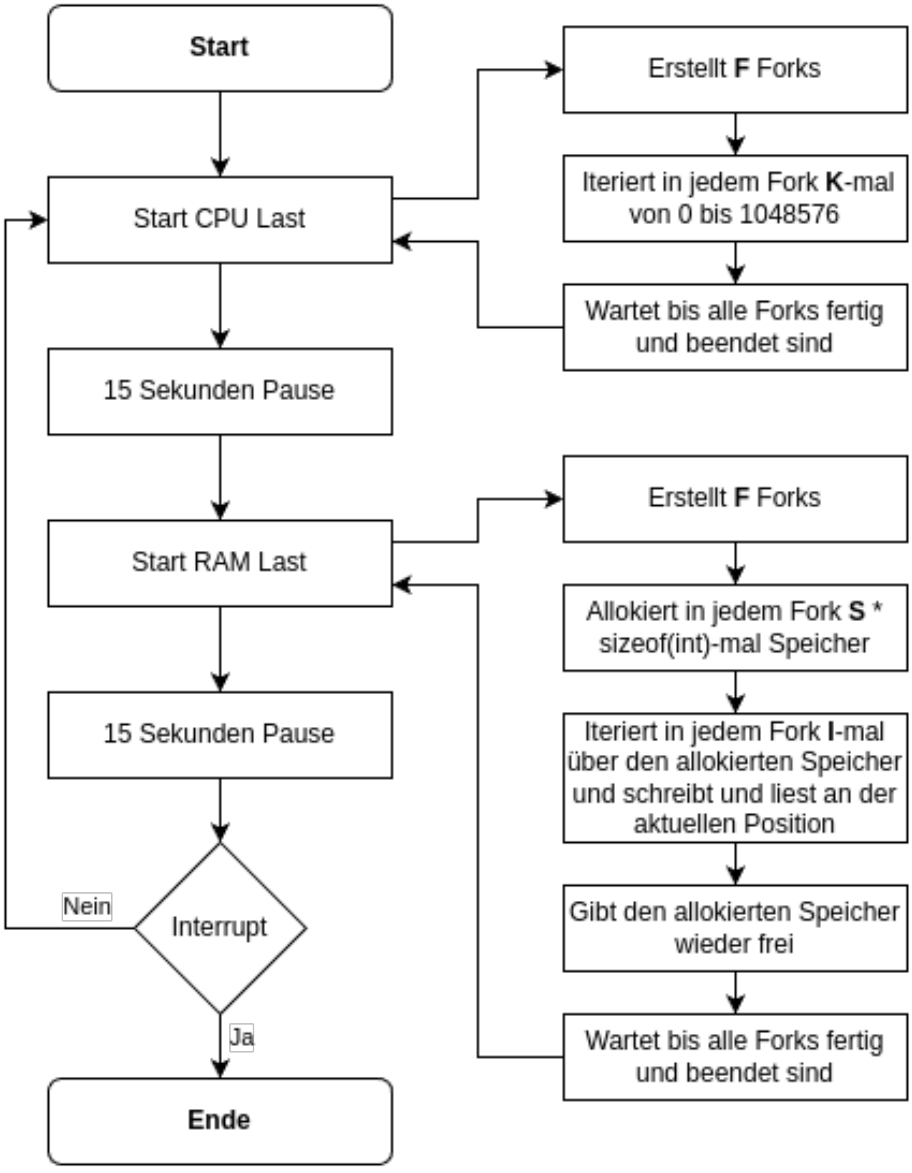

Abbildung 4.1: Programm Ablauf als Flussdiagramm

5 Messergebnisse

Im Folgenden werden durchgeführte Messungen und deren Charakteristik aufgezeigt und beschrieben.

5.1 Host itlx0028.hs-esslingen.de

Auf Host *itlx0028.hs-esslingen.de* wurde „Test Programm 1“ (siehe Kapitel 4.2 auf Seite 10) mit `F=60, K=80000, S=1048576, I=30000` ausgeführt. Dabei wurden Messungen mittels der RAPL und PowerCap durchgeführt.

Die dargestellten Messungen in Abbildung 5.1 auf Seite 13 und Abbildung 5.2 auf Seite 14 illustrieren die Auswirkungen auf den Energieverbrauch aus drei Zyklen.

Dabei korreliert der Energieverbrauch während den RAM-Zyklen mit dem Energieverbrauch der CPUs. Alle Prozesse werden beinahe zeitgleich mit den Zyklen fertig.

Abbildung 5.1: Messergebnisse itlx0028.hs-esslingen.de RAPL PACKAGE_ENERGY und DRAM_ENERGY (`F=60, K=80000, S=1048576, I=30000`)

5.2 Host worktower

Auf Host *worktower* wurde „Test Programm 1“ (siehe Kapitel 4.2 auf Seite 10) mit `F=20, K=30000, S=1048576, I=80000` ausgeführt. Dabei wurden Messungen mittels RAPL, PowerCap und dem Shelly Plug S durchgeführt.

Die dargestellten Messungen in Abbildung 5.3 auf Seite 14, Abbildung 5.4 auf Seite 15 und Abbildung 5.5 auf Seite 15 illustrieren die Auswirkungen auf den Energieverbrauch aus drei Zyklen.

In Abbildung 5.3 auf Seite 14 zeigen die Messungen von RAPL und PowerCap eine hohe Übereinstimmung. Die geringfügigen Unterschiede zwischen den Messwerten beider Schnittstellen liegen im Bereich von hundertstel Joule, was auf die unterschiedlichen Genauigkeiten der beiden Schnittstellen zurückzuführen ist. Die Messergebnisse zwischen dem Shelly Plug S, RAPL und PowerCap zeigen eine Korrelation. Mögliche Abweichungen können durch die Auslastung der Grafikkarte oder Festplatten entstehen, die durch andere

Abbildung 5.2: Messergebnisse itlx0028.hs-esslingen.de PowerCap ENERGY_UJ (Package) und SUBZONE_ENERGY_UJ (DRAM) (`F=60, K=80000, S=1048576, I=30000`)

Prozesse in Anspruch genommen werden. Dabei korreliert der Energieverbrauch während der RAM-Zyklen mit dem Energieverbrauch der CPUs. Der Energieverbrauch am Ende der Zyklen zeigt einen stufenweisen Rückgang, der auf das aktivierte Hyper-Threading zurückgeführt werden kann. Da die Prozesse nicht simultan auf der CPU ausgeführt werden können, resultiert dies in unterschiedlichen Laufzeiten.

Abbildung 5.4 auf Seite 15 und Abbildung 5.5 auf Seite 15 zeigen den Energieverbrauch von `Package` und `Powerplane 0`. Mittels der Abbildung 2.2 auf Seite 5 sind die Bereiche der unterschiedlichen Messpunkte aufgezeigt.

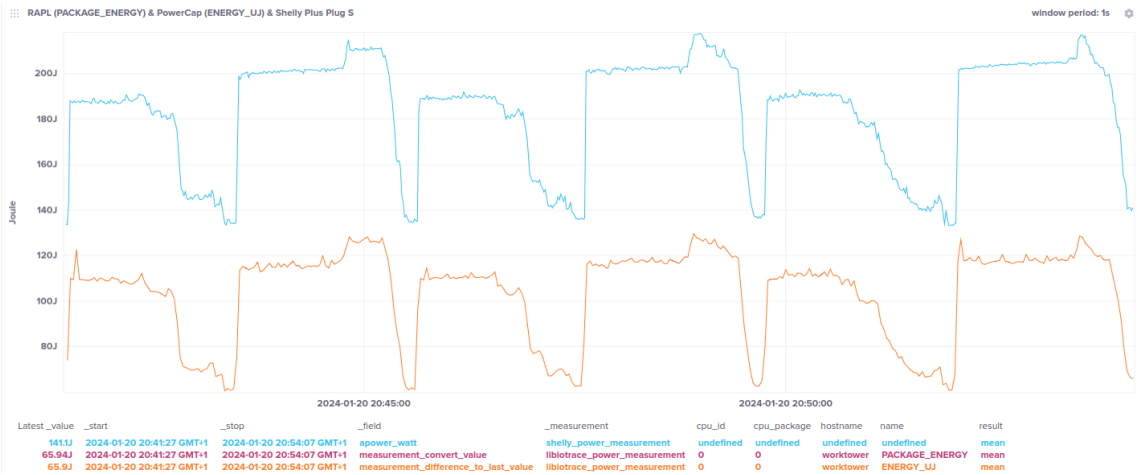

Abbildung 5.3: Messergebnisse worktower RAPL (PACKAGE_ENERGY), PowerCap (ENERGY_UJ) und Shelly Plus Plug S (`F=20, K=30000, S=1048576, I=80000`)

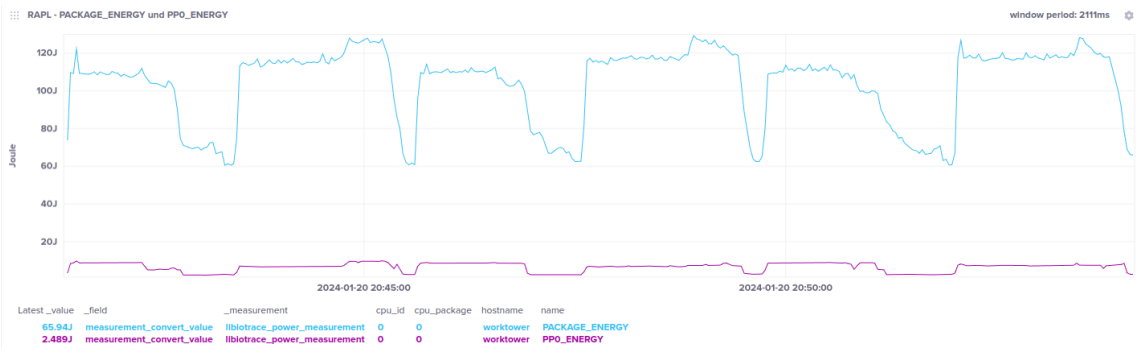

Abbildung 5.4: Messergebnisse worktower RAPL PACKAGE_ENERGY und PP0_ENERGY (`F=20, K=30000, S=1048576, I=80000`)

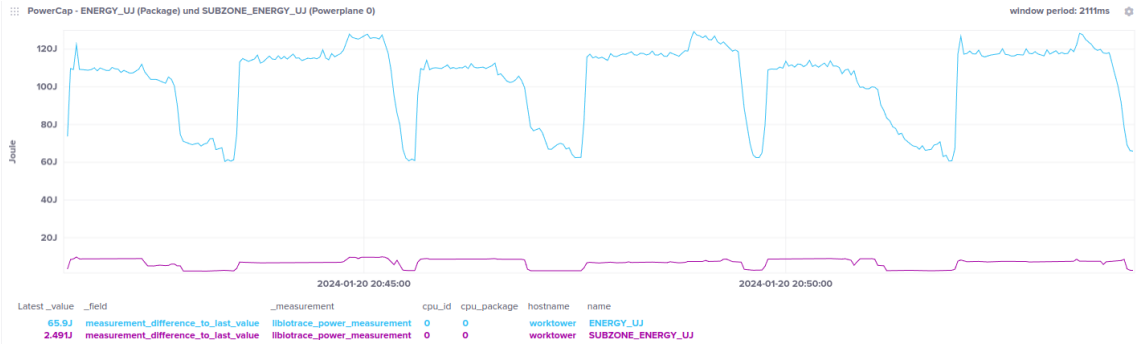

Abbildung 5.5: Messergebnisse worktower PowerCap ENERGY_UJ (Package) und SUBZONE_ENERGY_UJ (Powerplane 0) (`F=20, K=30000, S=1048576, I=80000`)

6 Fazit

Mein Ziel in Phase 1 des Forschungsprojekt bestand darin, die unterschiedlichen Schnittstellen der CPU zu evaluieren und aufzuzeigen, sowie die Erweiterung der „Libiotrace“ um das Erfassen von Live-Energiedaten.

Diese Ziele wurden durch die ersten Messungen aus Kapitel 5 auf Seite 13 erreicht. Die unterschiedlichen Schnittstellen wurden in Kapitel 2 auf Seite 3 erläutert und in Kapitel 3 auf Seite 6 zur „Libiotrace“ hinzugefügt.

Zu Beginn meiner Forschung habe ich das Portfolio um weitere `IO-Tests` erweitert. Ebenso habe ich unterstützt bei der Optimierung der Skripte zum Starten der Tests auf dem bwUniCluster. Im Zuge der Implementierung von RAPL und PowerCap entdeckte ich einen Fehler in der Kernfunktionalität von „Libiotrace“, der zu einem Performance Issue des Tools führte. Dieser Fehler wurde daraufhin von Philipp Köster übernommen und behoben.

7 Ausblick

In Phase 2 ist das Ziel die gesammelten Live-Energiedaten für die Entwickler aufzubereiten, sowie das Messen von Kennzahlen und die Optimierung dieser. Eine mögliche Kennzahl könnte hierfür der Datendurchsatz zu InfluxDB darstellen. Zum Aufbereiten der Daten könnte ein Dashboard im Grafana dienen.

Um die Erfassung von Live-Energiedaten weiter zu verbessern, müssen zusätzliche Testprogramme entwickelt werden, die gezielt Last, beispielsweise auf einen CPU-Sockel, erzeugen. Dies kann mittels der Message Passing Interface (MPI)-Technologie umgesetzt werden.

Messungen in einer „sauberen“ Umgebung, ohne andere aktive Prozesse auf dem System, wären ebenfalls wünschenswert. Hierfür könnte möglicherweise auf devDachs zurückgegriffen werden, auf dem wir ebenfalls Root-Zugriff besitzen.

8 Literaturverzeichnis

[1] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "Rapl in action: Experiences in using rapl for power measurements," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, no. 2, mar 2018. [Online]. Available: https://doi.org/10.1145/3177754

[2] S. G. PLC. [Online]. Available: https://www.shelly.com/de/products/shop/shelly-plus-plug-s-1

[3] A. Robotics. Switch. [Online]. Available: https://shelly-api-docs.shelly.cloud/gen2/ComponentsAndServices/Switch#switchgetstatus-example

[4] M. Walker, S. Bischoff, S. Diestelhorst, G. Merrett, and B. Al-Hashimi, "Hardware-validated cpu performance and energy modelling," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2018, pp. 44–53.

[5] J. May, "Mpx: Software for multiplexing hardware performance counters in multithreaded programs," in *Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001*, 2001, pp. 8 pp.–.

[6] V. M. Weaver, D. Terpstra, and S. Moore, "Non-determinism and overcount on modern hardware performance counter implementations," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 215–224.

[7] X. Wu and V. Taylor, "Utilizing hardware performance counters to model and optimize the energy and performance of large scale scientific applications on power-aware supercomputers," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2016, pp. 1180–1189.

[8] S. Desrochers, C. Paradis, and V. M. Weaver. A validation of dram rapl power measurements. [Online]. Available: https://web.eece.maine.edu/~vweaver/projects/rapl/2016_memsys_rapl.pdf

[9] W. Lin. Linux 5.8 brings per-core energy sensor support for amd cpus. [Online]. Available: https://linuxreviews.org/Linux_5.8_Brings_Per-Core_Energy_Sensor_Support_For_AMD_CPUs

[10] golem. Und nicht mehr dabei ist ... [Online]. Available: https://www.golem.de/news/linux-kernel-5-13-ist-fertig-2106-157696-3.html

[11] NIST. Cve-2020-12912. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-12912

[12] V. Weaver. Read the rapl registers on recent (>sandybridge) intel processors. [Online]. Available: https://web.eece.maine.edu/~vweaver/projects/rapl/rapl-read.c

[13] P. Francescon, S. Cora, and N. Satariano, "Calculation of for several small detectors and for two linear accelerators using monte carlo simulations," *Medical Physics*, vol. 38, no. 12, pp. 6513–6527, 2011. [Online]. Available: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.3660770

[14] A. Kogler, D. Weber, M. Haubenwallner, M. Lipp, D. Gruss, and M. Schwarz, "Finding and exploiting cpu features using msr templating," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 1474–1490.

[15] kernel.org. Power capping framework. [Online]. Available: https://www.kernel.org/doc/html/next/power/powercap/powercap.html

[16] J. Corbalan, L. Alonso, C. Navarrete, and C. Guillen, "Soft cluster powercap at supermuc-ng with ear," in *2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC)*, 2022, pp. 1–8.