

---

Gottfried Wilhelm Leibniz Universität Hannover  
Institut für Verteilte Systeme  
Distributed Computing & Security Group

Seminarausarbeitung  
im Studiengang Informatik (M.Sc.)

# Schriftliche Ausarbeitung für Seminar Aspekte Verteilte Systeme

Verfasser: Zhang, Zijian  
Betreuer: Dipl.-Inform. H. Tobaben  
Datum: 3. Juli 2016



# Inhaltsverzeichnis

<b>1 HPC</b>	<b>1</b>
1.1 Was ist HPC . . . . .	1
1.2 Warum HPC . . . . .	1
1.3 HPC und Energiebedarf . . . . .	1
<b>2 Versuchen</b>	<b>3</b>
2.1 Untersuchungsplattform . . . . .	3
2.2 Energiebedarf-relevante Aspekte . . . . .	4
2.2.1 Energiebedarf und Architektur . . . . .	4
2.2.2 Energiebedarf und Tasks . . . . .	5
2.2.3 Energybedarf und Maßstab von Tasks . . . . .	5
2.2.4 Energy bedarf und Implementationsmethode . . . . .	6
<b>3 Rückschlüsse und Ergänzungen</b>	<b>11</b>
<b>A Codes</b>	<b>13</b>



# Kapitel 1

## HPC

### 1.1 Was ist HPC

HPC(Auf Englisch High Performance Computing), oder Rechnen mit dem Hilfer von Superrechner, ist typischweise Supercomputer mit großen Anzahl der Prozessoren, die auf gemeinsame Peripheriegeräte und eine teilweise gemeinsamen Hauptspeicher zugreifen können, die die auf dem Wikipedia beschrieben werden.

### 1.2 Warum HPC

Um die Lösung maßstabreicher Probleme zu finden. Diese Probleme hat dem Merkmal, dass sie auf viele homogenisierende Teilprobleme bestanden, zum Beispiel die auf dem Bereich Hydromechanik, Biologie oder atmosphärische Wissenschaften. Um das Rechnungsprozess zu beschleunigen, ist parallele Rechnung sehr hilfreich. Das ist auch der Grund dafür, warum Supercomputers werden von Institute alle Land immer noch untersucht.

### 1.3 HPC und Energiebedarf

Unter der gleichem Architektur besitzt ein Supercomputer je mehr Prozessoren verbraucht es mehr Energie. Mit der Leistung von 93.000,00 TeraFLOPS beträgt der Energiebedarf von Sunway TaihuLight 15.370kW, was ist eigentlich zehr energieaufwändig. Aber dieser Zustand ist verbesserbar. Eine Seit durch den Fortschritten der Technik kann man die Architektur von Supercomputer monifizieren, um zum schluß die Energiebedarf per Prozessor zu senken, andere Seit ist die Softwareimpimentation auch verbesserbar. In diser Ausarbeitung biete ich ein paar Vergleichen an, der viele Energiebedarf-relevante Aspekte identifizieren.



# Kapitel 2

## Versuchen

### 2.1 Untersuchungsplattform

Unsere Untersuchung wurde durchgeführt auf dem Plattform wie unter:

Hardwaresystem	ODROID-XU3 Lab Environment(mit ARM Cortex-A7 1.4Ghz und Cortex-A15 2.0Ghz big.LITTLE architecture jeweils 4 kerne)
Betriebssystem	Ubuntu 15.10 mit ssh Zugriff und shared storage durch NFS server
Test-Benchmark	NAS Parallel Benchmarks
Tasks	LU Dekomposition und Gauß'sche Elimination(LU) und konjugierender Gradient(CG)
Task-Maßstab	siehe Tablett 2.1
Implementationssprache	Java
Implementationsmethode	OpenMP und MPI(Message Passing Interface)
Messungsgeräte	ODROID Smart Power Device

Tabelle 2.1: Untersuchungsplattform

		Class A	Class B
CG	Size	14000	75000
	Iteration	15	75
LU	Size	64x64x64	102x102x102
	Iteration	250	250

Tabelle 2.2: Task-Maßstab

## 2.2 Energiebedarf-relevante Aspekte

### 2.2.1 Energiebedarf und Architektur

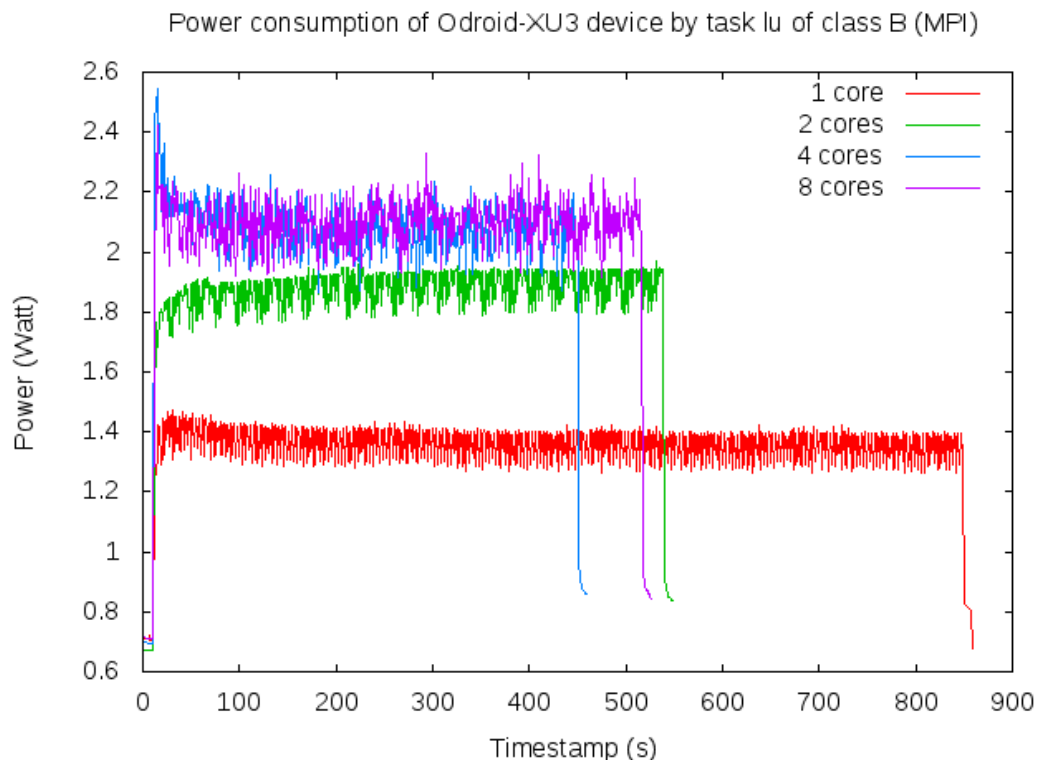


Abbildung 2.1: Leistung und Kernenutzung

Wie gezeigt in Abbildung 2.1 und sein Integral Abbildung 2.2. Die Rechnung befasst sich um das Task von LU und mit BWenn es nur ein Kern aktiviert ist, ist die Energiebedarf des Computersystems am höchsten und dauert die Rechnung am längsten. Deshalb die Konfiguration mit nur ein aktivierende Kern ist auf jeden Fall nicht effizient.

Aber es ist auch komisch, dass die Zeit- und Energiebedarf des Task von 8 Kerne sind beide höher als die von 4 Kerne. Laut der Dokumentation von ODROID-XU3 kann man feststellen, dass es 4 Kerne von Cortex-A7 und 4 Kerne von Cortex-A15 auf der Tafel sich befindet. Wenn es nur 4 Kerne aktiviert ist, sind die 4 Kerne mit niedrigere ID, d.h. 4 Kerne von Cortex-A7 lauffar.

Der Grunden dafür kann man mit dem Hilfe Dokumentationen und Versuchergebnisse vermuten. Rechnungsdatenumtauschung innerhalb ein CPU ist immer noch effizienter als die über 2 CPUs. Darum ist die Zeitbedarf mit 4 Kerne nideriger. Beschrieben von Webseite von ARM und in Beachtung von Messungsergebnis in Abbildung 2.3, ist die Leistung des Cortex-A15 höher als die des Cortex-A7 und für die Energiebedarf vice versa. Deshalb für eine Rechnungsarbeit mit so moderatem



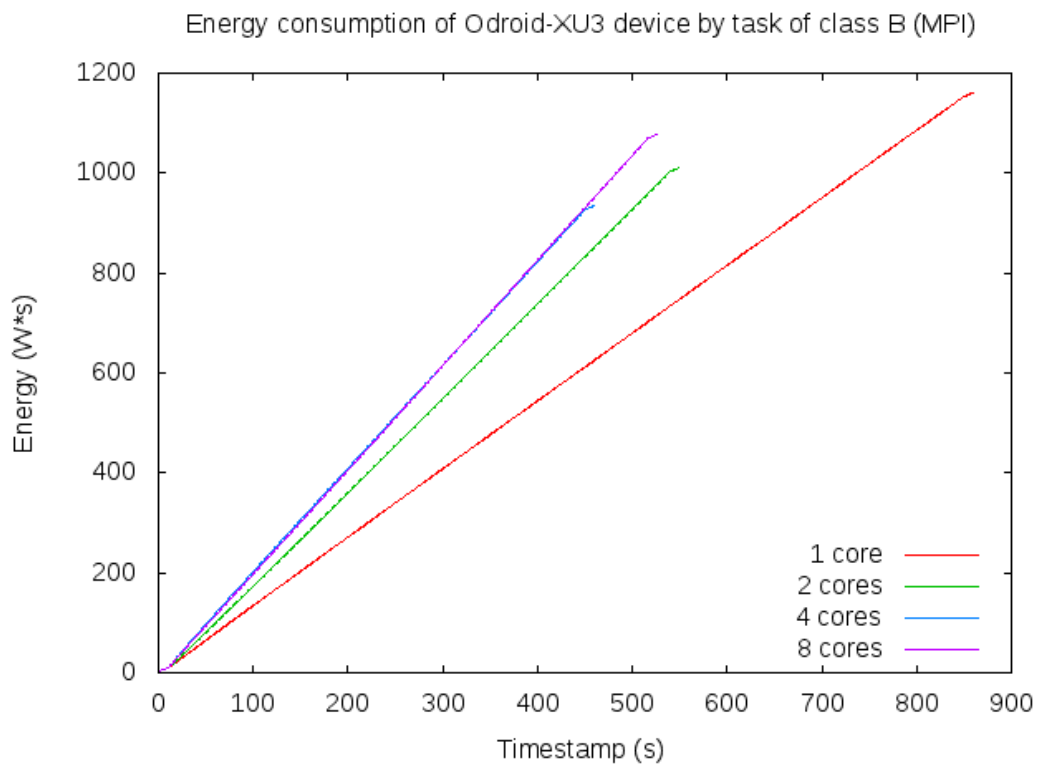


Abbildung 2.2: Energiebedarf und Kernenutzung

Maßstab soll man die energieeffiziente CPUs bevorzugen.

### 2.2.2 Energiebedarf und Tasks

Wie hergestellt in Abbildung 2.4 und ihre Integral 2.5, obwohl sich der gesamte Energiebedarf zwischen verschiedenen Tasks und Kernekonfigurationen sehr viel verändert, gibt es gar keinen Unterschied bezüglich Leistung dazwischen. Dadurch kann man vermuten, dass während der Versuchen befindet sich jedes CPU nur auf zwei Zustände: aktiviert oder nicht. Es gibt gar keine Zwischenzustand, auf dem ein CPU mit einer unterschiedlicher Leistung ausgeführt werden kann.

### 2.2.3 Energybedarf und Maßstab von Tasks

Die Abbildungen 2.6 und 2.7 zeigen, dass die Maßstäbe der Tasks nichts mit die Leistung des Durchgangs zu tun haben. Nur je größer die Tasks sind, braucht es mehr Zeit die Berechnung zu machen, desto wird mehr Energie insgesamt gebraucht.

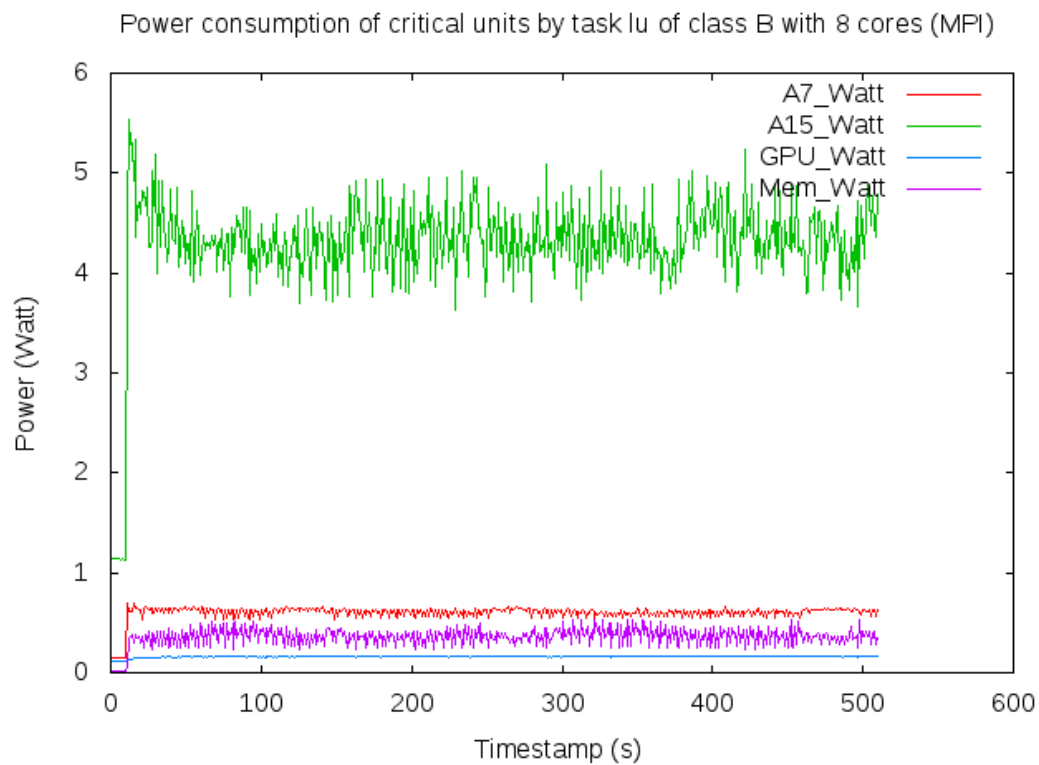


Abbildung 2.3: Leistung kritischer Teilen

#### 2.2.4 Energy bedarf und Implementationsmethode

Laut der Abbildungen 2.8 und ?? kann man sagen, dass für diese Versuchtasks spielt OpenMP eine bessere Rolle als MPI, und zusätzlich ist der Durchlauf mit 4 Kerne besser als der mit 8 Kerne. Der Grund dafür kann nicht nur das sein, das im 2.2.1 geschrieben werden hat, sondern auch wegen der Implementationsmethode. Weil ein lauffähige OpenMP-Programm während der Kompilierenphase in Parallelbereiche zugeordnet worden hat, jedes davon aus viele parallelausführbare Programmteil besteht. In der Durchlaufphase kann es sehr einfach in CPU-kerne zugeordnet. Aber die MPI-Programme brauch Kommunikation durch Peripherien, was eine höhere Zeit- und Energiebedarf verursacht.

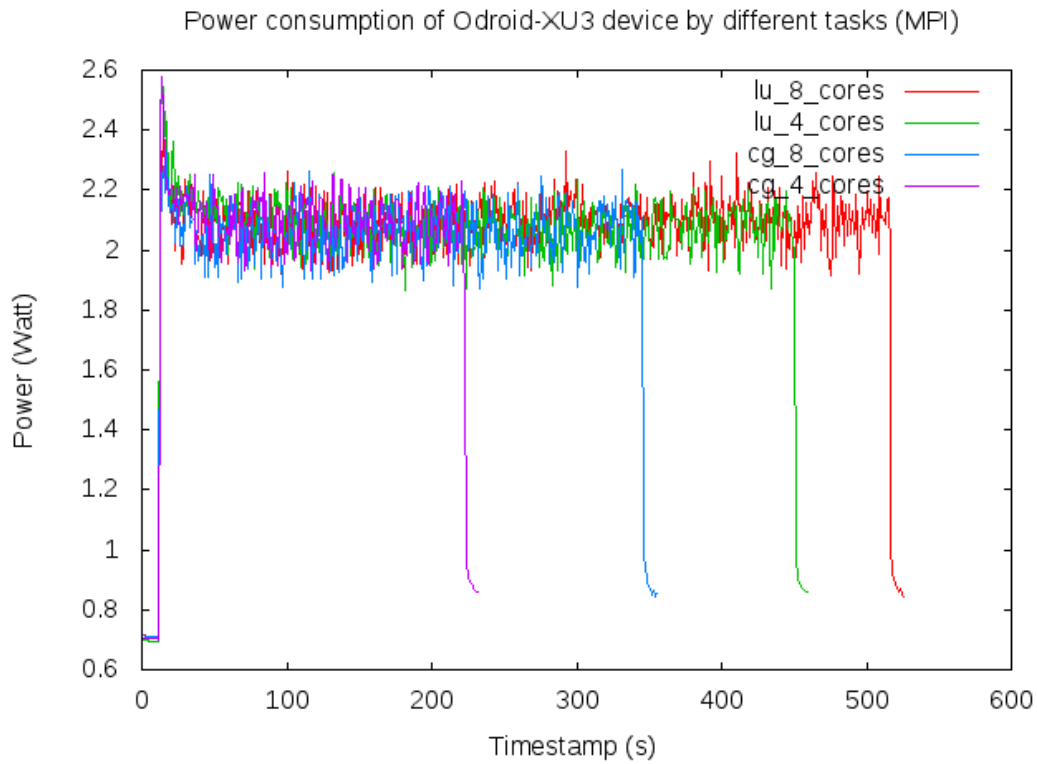


Abbildung 2.4: Leistung und Tasks

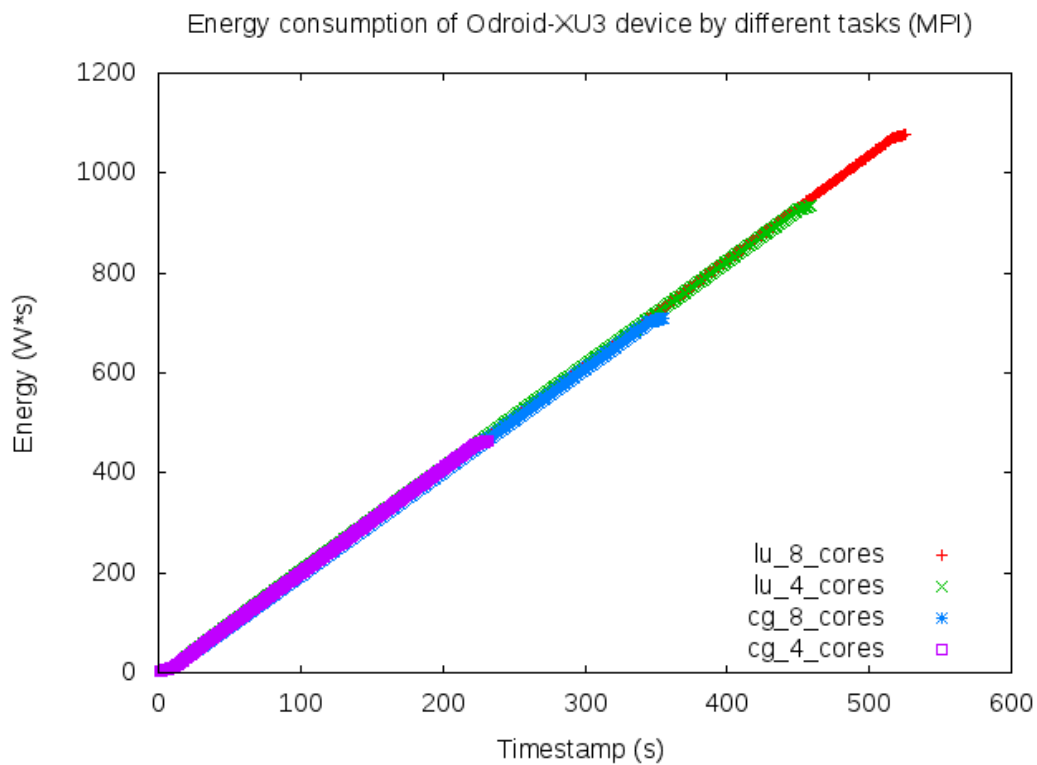


Abbildung 2.5: Energiebedarf und Tasks

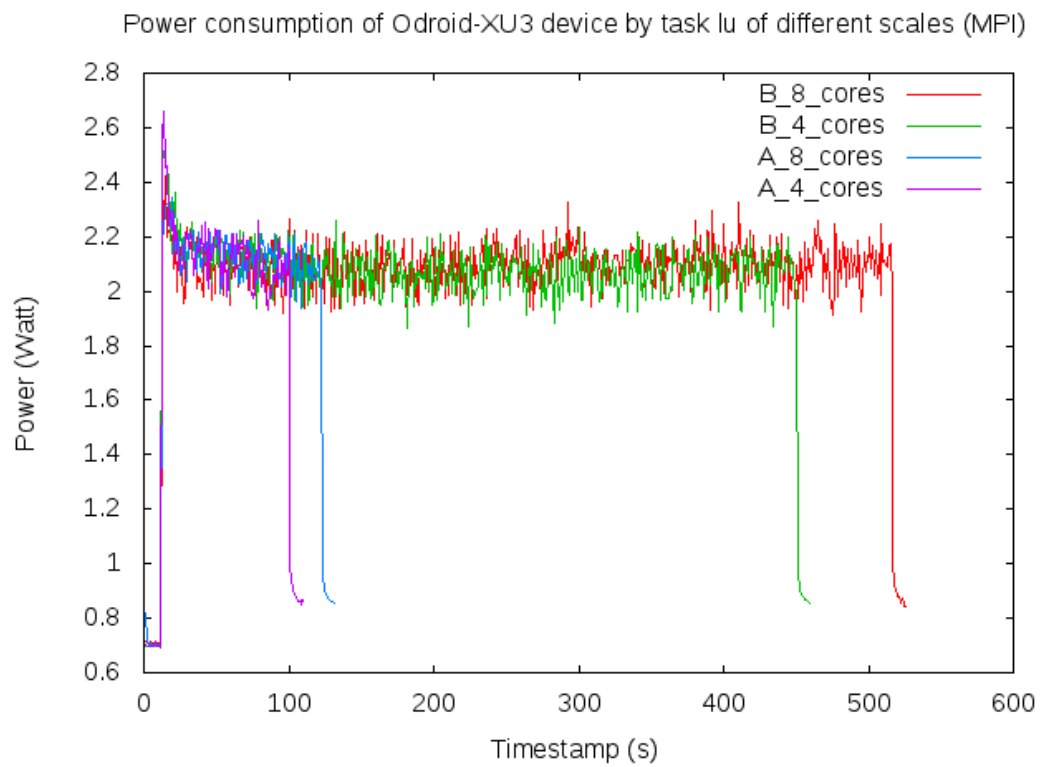


Abbildung 2.6: Leistung und Maßstab von Tasks

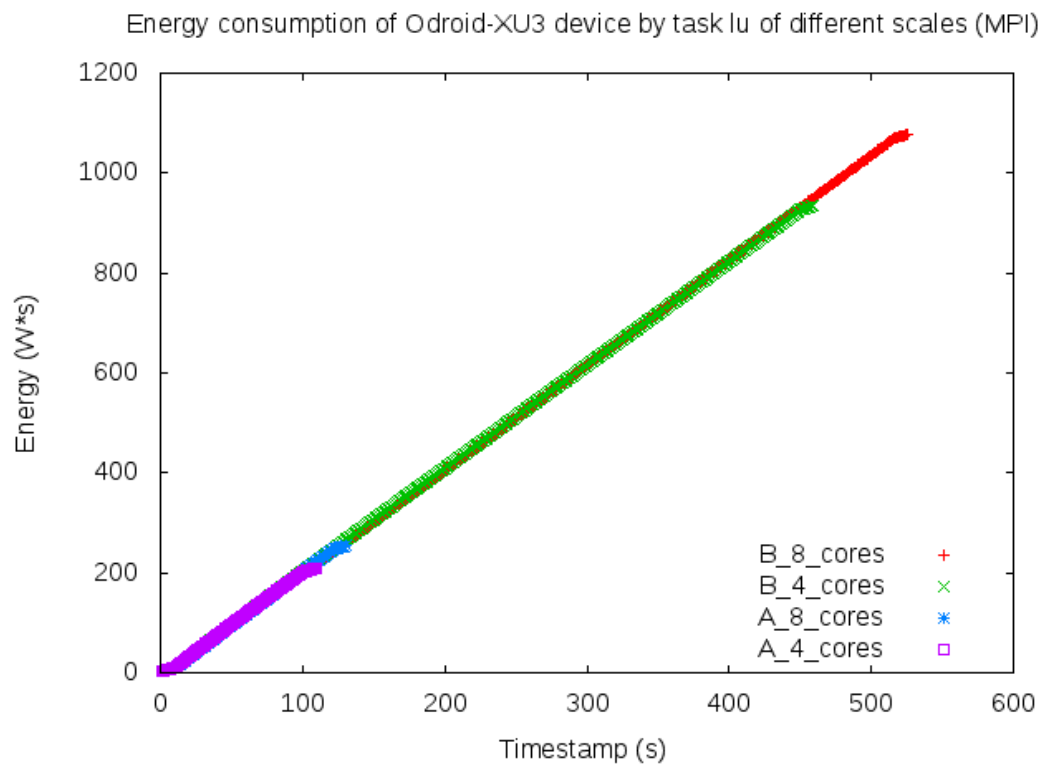


Abbildung 2.7: Energiebedarf und Maßstab von Tasks

Power consumption of Odroid-XU3 device by task lu of class B with different implementa

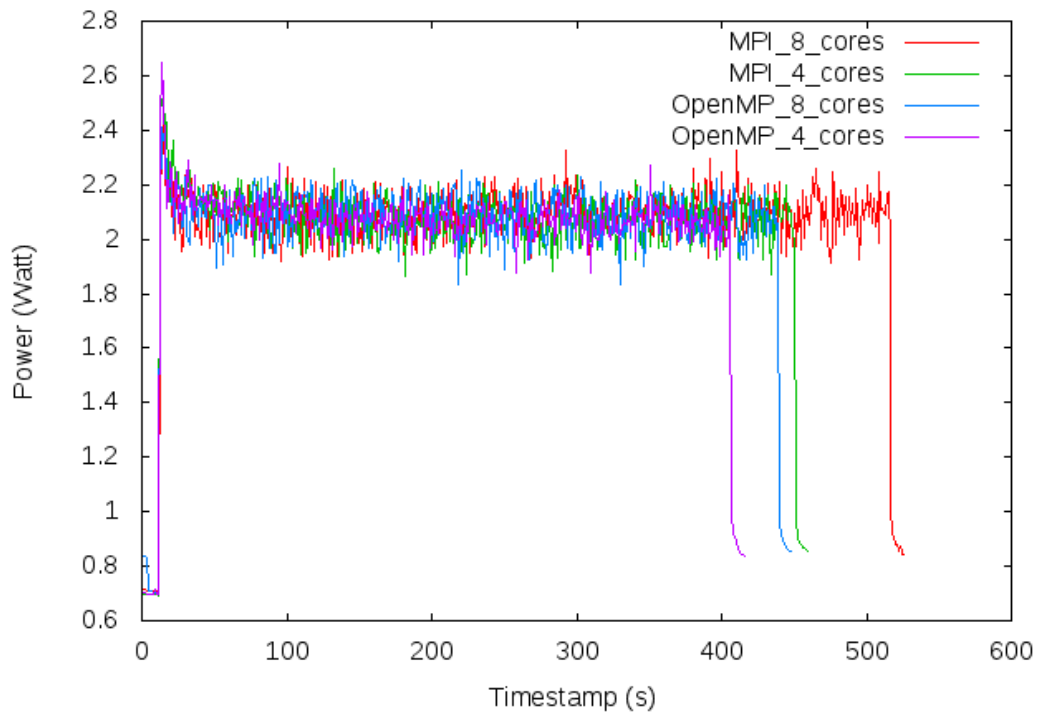


Abbildung 2.8: Leistung und Implementationsmethode

Energy consumption of Odroid-XU3 device by task lu of class B with different implement

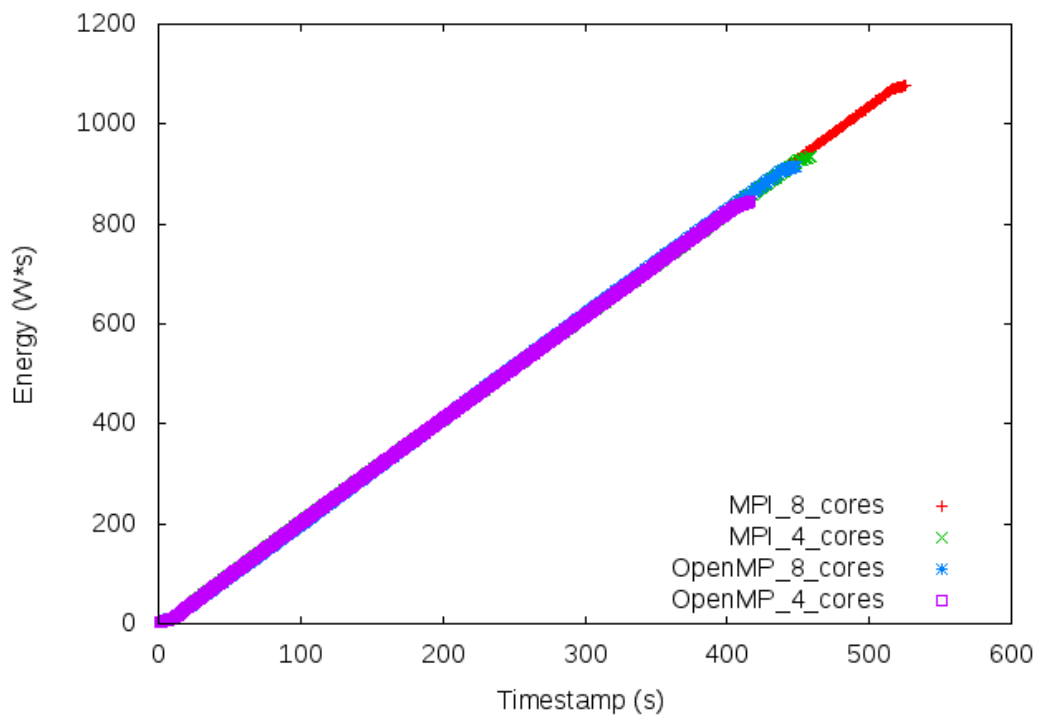


Abbildung 2.9: Energiebedarf und Implementationsmethode



## Kapitel 3

# Rückschlüsse und Ergänzungen

Der ganze Versuch stellt her, dass der Energiebedarf unter unsere Untersuchungsplattform stark abhängig von Anzahl der aktivierte CPUs und Implementationsmethode ist. Wir sollen vorsichtig diese Aspekte bezüglich unsere Task und Plattform feststellen. Es ist zur Ergänzungen, dass es für beide MPI und OpenMP, beziehungsweise für die CPUs sich selbe noch viele regelbare Parameteren gibt, zum Beispiel Environment Variable `OPM_NUM_THREADS` bestimmt die Anzahl von Threads zu aktivieren für einen Parallelbereich, und ebenfalls für mpi Ausführungsprogramm `mpirun.mpic` gibt es auch viele sachen die regelbar sind.





# Anhang A

## Codes

In diesem Kapitel werden alle automatisierte Skripte (zu kompilieren und messen, beziehungsweise für die Abbildungsherstellung) dargestellt

```
import os

result_dir = "~/res"
base_dir = "~/LAB/NAS/"
ver_dir = "NPB3.3.1/"
ver_short = "NPB3.3"
impls = ["MPI", "OMP"]

benchmark_names = ["CG", "LU"]
class_names = ["S", "W", "A", "B", "C", "D"]

for impl in impls:
    cp_command = "cp "+base_dir+impl+"/make.def "+
                  base_dir+ver_dir+ver_short+
                  "-"+impl+"/config/"

    os.system(cp_command)

    for bn in benchmark_names:
        for cn in class_names:
            for core_num in [1,2,4,8]:
                make_command = "cd "+base_dir+ver_dir+
                                ver_short+"-"+impl+
                                "/;make "+bn

                if impl == "MPI":
```

```
        make_command+=" NPROCS="+  
            str (core_num)  
  
        elif core_num>1:  
            break  
  
        make_command += " CLASS="+cn  
        os.system(make_command)
```

Listing A.1: compile.py

```
#!/bin/bash  
  
res_dir="/home/user5/res/"  
if [ ! -f $resultdir ];  
then  
    mkdir -p $resultdir  
fi  
  
basedir="/home/user5/LAB/NAS/"  
verdir="NPB3.3.1/"  
vershort="NPB3.3"  
  
impls="MPI OMP"  
benchmark_names="cg lu"  
class_names="A B"  
  
core_nums="1 2 4 8"  
  
mes_sensor="stdbuf -oL read-xu3-sensors -c -t"  
mes_power="stdbuf -oL smartpower -c -t"  
  
for impl in $impls;  
do  
    for bn in $benchmark_names;  
    do  
        for cn in $class_names;  
        do  
            for core_num in $core_nums;  
            do  
                res_folder=$res_dir "/" $impl "/"
```

---

```

if [ ! -f $res_folder ];
then
    mkdir -p $res_folder
fi

sensor_res_filename=$res_folder$bn ".\" \
    $cn ".\" $core_num \
    ". sensor.txt "
power_res_filename=$res_folder$bn ".\" \
    $cn ".\" $core_num \
    ". power.txt "

if [ -e $sensor_res_filename ] &&
    [ 'wc -l $sensor_res_filename |
        cut -f 1 --delimiter=" " ' != "0" ] &&
    [ -e $power_res_filename ] &&
    [ 'wc -l $power_res_filename |
        cut -f 1 --delimiter=" " ' != "0" ];
then
    continue
fi

($mes_sensor)>$sensor_res_filename&
pid_mes_sensor=$!
($mes_power)>$power_res_filename&
pid_mes_power=$!

sleep 10

cd "/home/user5/LAB/NAS/NPB3.3.1/NPB3.3-"$impl"/ bin "
if [ $impl == "MPI" ];
then
    run_cmd="mpirun.mpich -np $core_num ./\" \
        $bn ".\" $cn ".\" $core_num
else
    export OMP_NUM_THREADS=$core_num
    run_cmd="./\" $bn ".\" $cn ".x"
fi

```

```
($run_cmd)
sleep 10

kill $pid_mes_sensor
kill $pid_mes_power

sync
sleep 2
done
done
done
done
```

Listing A.2: measure.sh

```
set term png
unset log
unset label
set xtic auto
set ytic auto
set datafile separator ","
set key right top

set title "Power consumption of Odroid-XU3 device by\
          task lu of class B (MPI)"
set xlabel "Timestamp (s)"
set ylabel "Power (Watt)"
set output "cores_power.png"
plot\
    "MPI/lu.B.1.power.txt" u 0:3 t "1 core" with line ,\
    "MPI/lu.B.2.power.txt" u 0:3 t "2 cores" with line ,\
    "MPI/lu.B.4.power.txt" u 0:3 t "4 cores" with line ,\
    "MPI/lu.B.8.power.txt" u 0:3 t "8 cores" with line ;

set title "Power consumption of critical units by task\
          lu of class B with 8 cores (MPI)"
set output "units_power.png"
plot\
    "MPI/lu.B.8.sensor.txt" u 0:4 t "A7_Watt" with line ,\
    "MPI/lu.B.8.sensor.txt" u 0:7 t "A15_Watt" with line ,\
```

```

"MPI/lu.B.8.sensor.txt" u 0:10 t "GPU_Watt" with line ,\
"MPI/lu.B.8.sensor.txt" u 0:13 t "Mem_Watt" with line

set title "Power consumption of Odroid-XU3 device by\
          different tasks (MPI)"
set output "tasks_power.png"
plot\
    "MPI/lu.B.8.power.txt" u 0:3 t "lu_8_cores" with line ,\
    "MPI/lu.B.4.power.txt" u 0:3 t "lu_4_cores" with line ,\
    "MPI/cg.B.8.power.txt" u 0:3 t "cg_8_cores" with line ,\
    "MPI/cg.B.4.power.txt" u 0:3 t "cg_4_cores" with line ;

set title "Power consumption of Odroid-XU3 device by\
          task lu of different scales (MPI)"
set output "scales_power.png"
plot\
    "MPI/lu.B.8.power.txt" u 0:3 t "B_8_cores" with line ,\
    "MPI/lu.B.4.power.txt" u 0:3 t "B_4_cores" with line ,\
    "MPI/lu.A.8.power.txt" u 0:3 t "A_8_cores" with line ,\
    "MPI/lu.A.4.power.txt" u 0:3 t "A_4_cores" with line ;

set title "Power consumption of Odroid-XU3 device by\
          task lu of class B with different implementations"
set output "implementations_power.png"
plot\
    "MPI/lu.B.8.power.txt" u 0:3 t "MPI_8_cores" with line ,\
    "MPI/lu.B.4.power.txt" u 0:3 t "MPI_4_cores" with line ,\
    "OMP/lu.B.8.power.txt" u 0:3 t "OpenMP_8_cores"\
    with line ,\
    "OMP/lu.B.4.power.txt" u 0:3 t "OpenMP_4_cores"\
    with line ;

set key right bottom
set xlabel "Timestamp (s)"
set ylabel "Energy (W*s)"

```

```
set title "Energy consumption of Odroid-XU3 device by task\
        of class B (MPI)"
sum1=0; sum2=0; sum4=0;sum8=0;
set output "cores_energy.png"
plot\
    "MPI/lu.B.1.power.txt" u 0:(sum1=sum1+$3) t "1 core"\
    with line ,\
    "MPI/lu.B.2.power.txt" u 0:(sum2=sum2+$3) t "2 cores"\
    with line ,\
    "MPI/lu.B.4.power.txt" u 0:(sum4=sum4+$3) t "4 cores"\
    with line ,\
    "MPI/lu.B.8.power.txt" u 0:(sum8=sum8+$3) t "8 cores"\
    with line ;
```

```
set title "Energy consumption of Odroid-XU3 device by\
        different tasks (MPI)"
sum1=0; sum2=0; sum4=0;sum8=0;
set output "tasks_energy.png"
plot\
    "MPI/lu.B.8.power.txt" u 0:(sum2=sum2+$3)\
    t "lu_8_cores",\
    "MPI/lu.B.4.power.txt" u 0:(sum1=sum1+$3)\
    t "lu_4_cores",\
    "MPI/cg.B.8.power.txt" u 0:(sum8=sum8+$3)\
    t "cg_8_cores",\
    "MPI/cg.B.4.power.txt" u 0:(sum4=sum4+$3)\
    t "cg_4_cores";
```

```
set title "Energy consumption of Odroid-XU3 device by\
        task lu of different scales (MPI)"
sum1=0; sum2=0; sum4=0;sum8=0;
set output "scales_energy.png"
plot\
    "MPI/lu.B.8.power.txt" u 0:(sum8=sum8+$3) t "B_8_cores",\
    "MPI/lu.B.4.power.txt" u 0:(sum4=sum4+$3) t "B_4_cores",\
    "MPI/lu.A.8.power.txt" u 0:(sum2=sum2+$3) t "A_8_cores",\
    "MPI/lu.A.4.power.txt" u 0:(sum1=sum1+$3) t "A_4_cores";
```

```

set title "Energy consumption of Odroid-XU3 device by\
          task lu of class B with different implementations"
sum1=0; sum2=0; sum4=0;sum8=0;
set output "implementations_energy.png"
plot\
    "MPI/lu.B.8.power.txt" u 0:(sum8=sum8+$3)\
    t "MPI_8_cores",\
    "MPI/lu.B.4.power.txt" u 0:(sum4=sum4+$3)\
    t "MPI_4_cores",\
    "OMP/lu.B.8.power.txt" u 0:(sum2=sum2+$3)\
    t "OpenMP_8_cores",\
    "OMP/lu.B.4.power.txt" u 0:(sum1=sum1+$3)\
    t "OpenMP_4_cores";

set title "Processing units' temperature monitoring on\
          running lu class B with 8 cores (MPI)"
set xlabel "Timestamp (s)"
set ylabel "Temperature (Celsius Degree)"
set output "mpi_temp.png"
plot\
    "MPI/lu.B.8.sensor.txt" u 0:23 t "CPU4_Temp" with line ,\
    "MPI/lu.B.8.sensor.txt" u 0:24 t "CPU5_Temp" with line ,\
    "MPI/lu.B.8.sensor.txt" u 0:25 t "CPU6_Temp" with line ,\
    "MPI/lu.B.8.sensor.txt" u 0:26 t "CPU7_Temp" with line ,\
    "MPI/lu.B.8.sensor.txt" u 0:26 t "GPU_Temp" with line ;

set title "Processing units' temperature monitoring on\
          running lu class B with 8 cores (OpenMP)"
set output "openmp_temp.png"
plot\
    "OMP/lu.B.8.sensor.txt" u 0:23 t "CPU4_Temp" with line ,\
    "OMP/lu.B.8.sensor.txt" u 0:24 t "CPU5_Temp" with line ,\
    "OMP/lu.B.8.sensor.txt" u 0:25 t "CPU6_Temp" with line ,\
    "OMP/lu.B.8.sensor.txt" u 0:26 t "CPU7_Temp" with line ,\
    "OMP/lu.B.8.sensor.txt" u 0:26 t "GPU_Temp" with line ;

```

Listing A.3: draw.p