**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Дисциплина:**

«Операционные системы»

**Лабораторная работа No3**

“Linpack”

Выполнил:

Хоанг Вьет Зунг – Студент группы N32491

Нгуенг Хонг Ньят - Студент группы N32451

Преподаватель:

Савков С.В.



Санкт-Петербург 2023

**Задание:**

Простой вариант:

Найти и скомпилировать программу linpack для оценки производительности

компьютера (Flops) и протестировать ее при различных режимах работы ОС:

1. С различными приоритетами задачи в планировщике

2. С наличием и отсутствием привязки к процессору

3. Провести несколько тестов, сравнить результаты по 3 сигма или другим

статистическим критериям

Усиленный вариант

То же самое, плюс изменить параметры на уровне ядра (выбрать одно):

1. Запретить выполнение всех потоков кроме того, который тестируется (путем

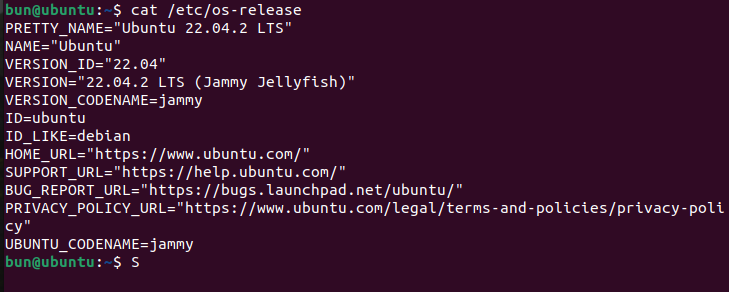
запрета прерываний) (cli sti)

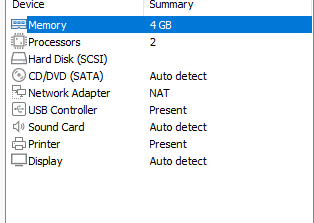
2. Найти другие планировщики процессов для 1Linux и сравнить результаты

работы вычислительной задачи на них

3. Повлиять на настройки имеющегося планировщика

4. Вмешаться в работу планировщика на уровне ядра

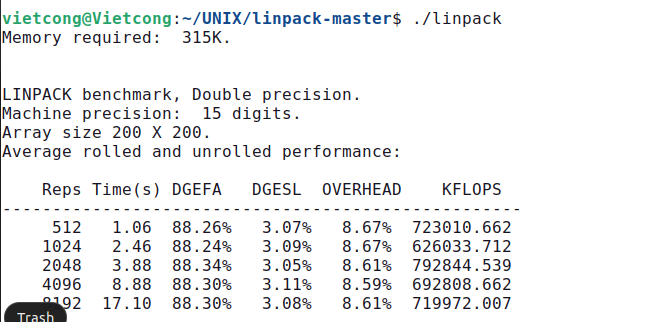
**Name: Ubuntu** 

****

**1. Простой вариант:**

- С различными приоритетами задачи в планировщике

Запустить с стандартными приоритетами:

****

The result of running the Linpack benchmark is typically reported as a performance rating in units of floating-point operations per second (FLOPS). The performance rating reflects the raw computing power of the computer system in solving large systems of linear equations using the LU factorization method.

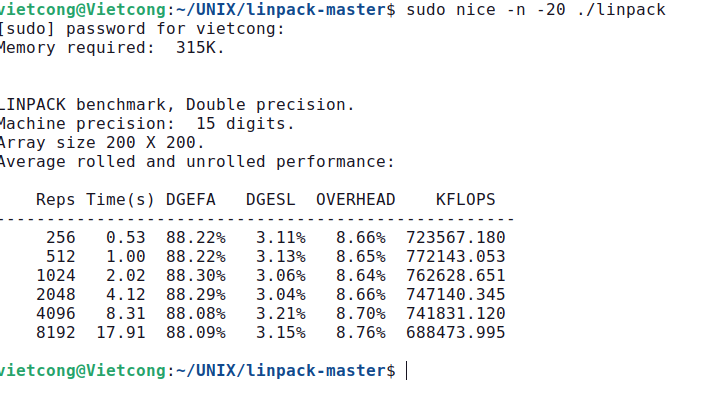
* Memory required: 315K: This indicates the amount of memory required by the benchmark in kilobytes (K).
* LINPACK benchmark, Double precision: This indicates that the benchmark is using the LINPACK benchmark suite and is using double precision floating-point arithmetic.
* Machine precision: 15 digits: indicates the precision of the floating-point arithmetic used by the benchmark in terms of the number of significant digits.
* Array size 200 X 200: indicates the size of the matrix used in the benchmark in terms of the number of rows and columns.
* Reps: indicates the number of repetitions used in the benchmark for each matrix size.
* Time(s): indicates the time it took to complete the benchmark for each matrix size in seconds.
* DGEFA: indicates the percentage of time spent in the LU factorization step of the benchmark.
* DGESL: indicates the percentage of time spent in the solution step of the benchmark.
* OVERHEAD: indicates the percentage of time spent in overhead tasks such as initialization and memory allocation.
* KFLOPS: indicates the performance rating of the system in terms of thousands of floating-point operations per second (KFLOPS) achieved during the benchmark.

For example, for an array size of 200 x 200 and 512 repetitions, the benchmark took 1.06 seconds to complete. The LU factorization step took 88.26% of the total time, while the solution step took 3.07% of the total time. The overhead tasks took 8.67% of the total time. The system achieved a performance rating of 723010.662 KFLOPS.

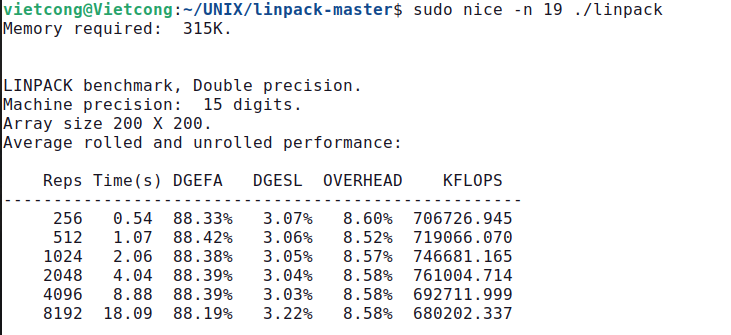
We use nice command to run the program with a specific scheduling priority. On a Linux system, the nice value ranges from -20 (highest priority) to 19 (lowest priority).

The **nice** command is a Linux utility that allows us to adjust the priority of a process. By default, processes are run with a priority level of 0, but you can use the **nice** command to lower or raise the priority of a process. The lower the priority level, the less CPU time the process is given.

The command: “sudo nice -n -20 ./linpack” sets the priority of the **./linpack** process to the highest possible level (-20), which means that the process will be given the highest priority for CPU time. This means that the system will allocate the most resources to the Linpack program, helping to increase computation speed and reduce latency. This can be useful if we want the LINPACK benchmark runs as quickly and efficiently as possible, without being slowed down by other processes running on the system.



The command: “sudo nice -n 19 ./linpack” sets the **./linpack** process to the lowest priority for CPU time. This can be useful if we want to ensure that other processes running on the system are given priority access to the CPU resources, and want to avoid the benchmark interfering with their performance.



There are no programs running on the computer other than linpack, performance with priority -20 is better than with 19.

- С наличием и отсутствием привязки к процессору

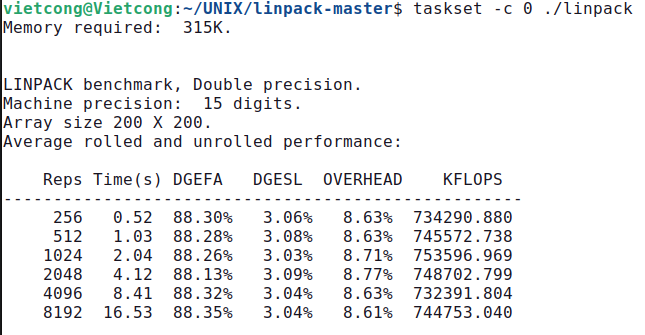
При отсутствии привязки к ядру производительность такая же, как и выше.

При наличие привязки к ядру: Использовать команду taskset. Команда taskset — установить или получить привязку процессора к процессу.

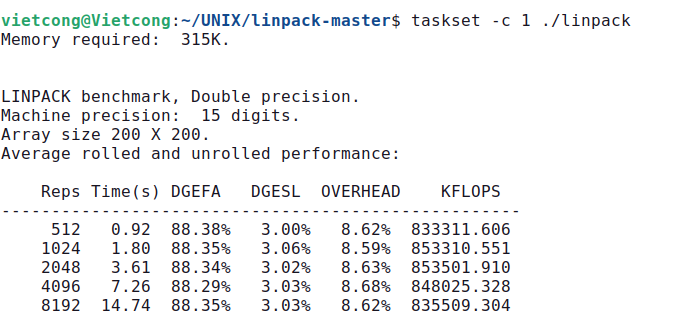
In case of a kernel binding, we use the taskset command. By default, a process can run on any available CPU core in the system. The taskset command is to set or get the processor affinity for a process.

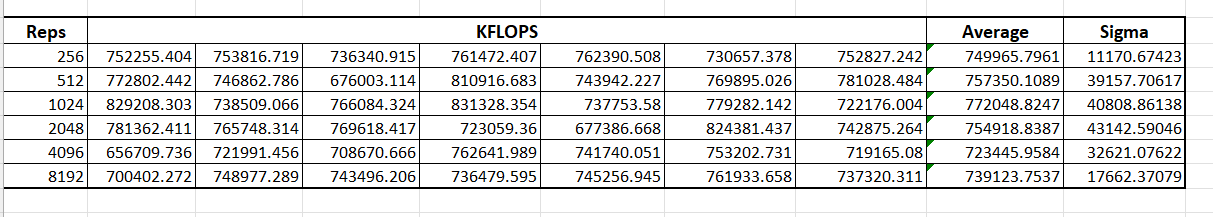
The taskset command is used to assign one or more CPUs or cores to a specific process. Assigning specific CPU cores to a process can have a significant impact on the benchmark results, as it can affect the way that the CPU caches are used by the process, and how the CPU interacts with other components of the system. In some cases, assigning specific cores to a benchmark process can result in improved performance, as the process can take advantage of specific features of those cores.

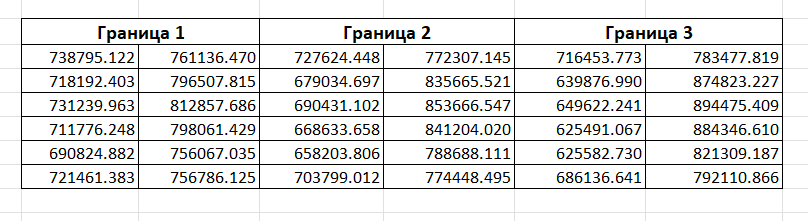
In this command, the option "-c 0" is used to specify that the linpack program runs only on CPU core 0, we are explicitly directing the linpack program to run on a specific CPU core, which can help to improve performance and ensure that the program is not interrupted or slowed down by other processes running on the system.



In this command, the option "-c 1" is used to specify that the linpack program runs only on CPU core 1.



- Провести несколько тестов, сравнить результаты по 3 сигма или другим статистическим критериям:

We see that all values are in the interval (cp-3sigma; cp+3sigma).

**2. Усиленный вариант**

Повлиять на настройки имеющегося планировщика

Начальные настройки:

Graphical user interface, text, application

Description automatically generated

The command "sudo sysctl -A | grep "sched" | grep -v "domain"" is used to display a list of system parameters related to process scheduling in Linux, while filtering out any parameters related to CPU frequency scaling.

- "sysctl -A": displays all system parameters and their values, with one parameter per line.

- "| grep "sched"": pipes the output of the previous command to the grep command, which searches for lines containing the word "sched". This filters the output to show only the system parameters related to process scheduling.

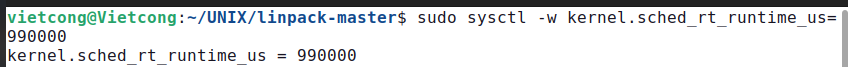
- "| grep -v "domain"": pipes the output of the previous command to another grep command, which searches for lines that do not contain the word "domain". This filters out any system parameters related to CPU frequency scaling, since these are not directly related to process scheduling.

До изменения:

Table

Description automatically generated with low confidence

Изменить значение sched\_rt\_runtime\_us:



The command "sudo sysctl -w kernel.sched\_rt\_runtime\_us= 990000" is used to set the maximum amount of time that real-time tasks are allowed to run in microseconds (us) on the system.

- "sysctl": This command is used to view, set, and modify kernel parameters at runtime.

- "-w": specifies that we want to write a new value to the specified kernel parameter.

- "kernel.sched\_rt\_runtime\_us": the name of the kernel parameter we want to modify. It specifies the maximum amount of time that real-time tasks are allowed to run on the system.

- " 990000": specifies the new value we want to set for the "kernel.sched\_rt\_runtime\_us" parameter. In this case, we are setting it to 990000 microseconds (us), which is equivalent to 990 milliseconds (ms).

Настройки после изменения:

Text

Description automatically generated

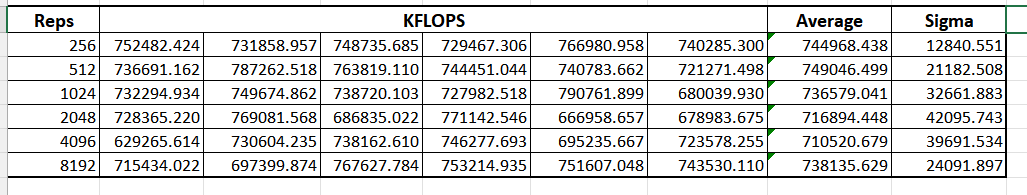
linpack после изменения:

Text, table

Description automatically generated

- Провести несколько тестов, сравнить результаты по 3 сигма или другим статистическим критериям

**After some tryouts, I noticed that in general, increasing the parameter from 950000 to 990000 will allow real-time tasks to run for a longer time period. Since I’ve been running only linpack and there weren’t many other real-time tasks, the KFLOPS parameters slightly increased which indicates better performance. But on the other hand, if the system is already struggling to handle the workload (heavily loaded with real-time tasks), increasing the parameter can decrease the system's performance and since it reduces the amount of CPU time available for other tasks.**



Table

Description automatically generated

**3. Вывод**

- В ходе данной работы, с помощью linpack я научился работать для получения информации о мощности вычислительной машины.

- Также, в усложненном варианте, научился работать с планировщиком, с тех пор мы можем более эффективно управлять и использовать вычислительные ресурсы.