

**Софийски Университет „Св. Климент Охридски”**

**Факултет по математика и информатика**

**Курсов проект**

**По Разпределени софтуерни архитектури**

**Задача:** *Brute Forcer*

**Изготвил: Даниел Константинов Кацаров, фак.№: 61600, 3 курс, Софтуерно инженерство**

**Ръководител: ас. Христо Христов**

**Дата: 27.06.2015г. Проверил: ………………….. (подпис)**

1. **Условие на задачатa.**

Задачата е да се реализира платформа за паралелна обработка на алгоритми и задачи, където основната нишка играе ролята на разпределител на задачи, а работните нишки чакат да им бъде възложена задача. При завършването на отделните задачи, работните нишки предават резултатите на нишка, която обработва асинхронните данни и извиква предефинирани хендлъри, които да комбинират резултатите и изведат крайния резултат.

В конкретния случай, за да докажем функционалността на системата ще изпълним алгоритъм за генериране на пароли от зададена азбука и дълбочина.

***Изискванията към програмата са следните***:

1. Програмата е реализирана на C, като са използвани блокваща опашка и синхронизационни инструменти (mutexes, conditional variables).
2. Програмата предоставя интерпретатор в реално време, чрез който потребителя може да избере броя на нишките, които ще обработват алгоритъма.
3. Програмата извежда подходящи съобщения на различните етапи от работата си.
4. **Описание на реализацията.**

Проекта е реализиран на С, като са използвани следните инструменти за многонишкова синхронизация:

* Мутекси за гарантиране на уникален достъп до данните.
* Conditional variables – за синхронизиране (блокиране/разблокиране) на работните нишки.
* Споделена блокиращa oпaшка, която позволява синхронизацията на разпределянето на задачи, както и на събирането на асихнронно идващите резултати от различните работни нишки.

При стартиране на системата, основната нишка изпълнява следната последователност от действия:

* Инициализират се блокиращите опашки, като се заделят буфери с желания капацитет на опашките.
* Динамично се заделя памет за масив, който съдържа информацията за всяка работна нишка. Информацията се запазва под формата на структура на езика С. Тя съдържа идентификация на нишката, указатели към текуща задача, структура с аргументи за текущата задача, персонален мутекс, флаг за заетост, cond variable за синхронизация при делегирането на задачи.
* Изпълнява се цикъл, който стартира работните нишки и като аргумент им се подава структура с инструментите за конкретната нишка.
* Стартира се разпределителната нишка, която получава като аргумент целия масив от структури за отделните работни нишки.
* Стартира се нишката, която събира резултатите от изпълнението на отделните задачи.

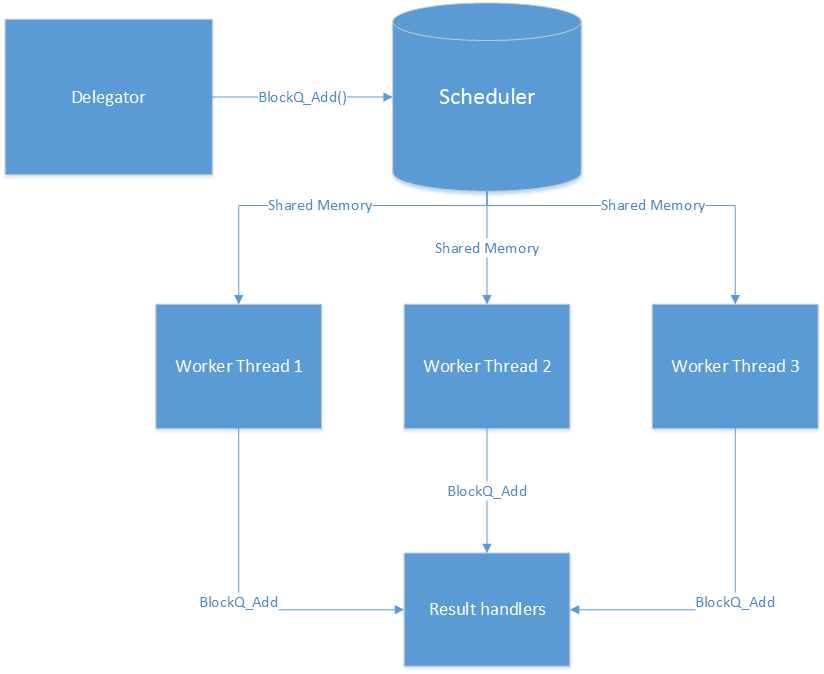
Основната последователност от действия при стартиране на системата е следния:

* След инициализация на всички инструменти, основната нишка задава конкретните подзадачи, в нашия случай генериране на пароли от определена азбука с определена дълбочина, чрез подготвяне на структурата с указател към функцията и аргументите и, и извикване на функцията за добавяне на елемент в опашката.
* Разпределителната нишка се разболкира и започва процес на откриване на 1-вата свободна работна нишка, на която да делегира извлечената задача
* Свободната работна нишка бива разблокирана и получава указател към функция и аргументи. Нишката извиква подадената функция и след завършване попълва върнатата стойност от функцията в структура която праща като елемент в блокиращата опашка на нишката, която събира резултатите от изпълнението.

Системата не разпаралелява конкретен алгоритъм, а дефинира обща платформа за паралелно обработване на множество задачи. Ускорението зависи от конкретния алгоритъм и от независимостта на подзадачите.

При алгоритми с ниска зависимост между подзадачите системата гарантира минимални разходи на процесорно време за синхронизацията на задачите, тъй като използва инструменти за синхронизация на ниско ниво.

***Диаграмта представлява компонентната декомпозиция на архитектурата на ниво паралелни процеси.***



1. **Резултати от измерването на ефективността, ускорението и времето**

|  |  |  |  |
| --- | --- | --- | --- |
| **Threads** | **Time** | **Acc** | **Efficiency** |
| **1** | **40** | **1** | **1** |
| **2** | **21** | **1.8** | **0.9** |
| **3** | **18** | **2.2** | **0.73** |
| **4** | **13** | **3** | **0.75** |
| **5** | **9** | **4.4** | **0.8** |
| **6** | **9** | **4.4** | **0.7** |
| **7** | **9** | **4.4** | **0.62** |
| **8** | **8** | **5** | **0.62** |
| **9** | **8** | **5** | **0.55** |
| **10** | **5** | **8** | **0.8** |
|  |  |  |  |