

# Изследване на скалируемостта на Wa-Tor симулацията при статична декомпозиция на домейна

Иван-Асен Веселинов Чакъров  
ФН: 81837, Курс: 3, Група: 1

Юли 2021

## 1 Увод

Wa-Tor [1] е класически проблем при паралелното програмиране. Накратко проблемът е симулирането на идеализиран двумерен свят с формата на тор. Светът има два вида обитатели - херинги, които играят ролята на плячка и акули, които ловят и изяждат херингите. Подробните правилата за симулацията са дадени във [1].

Декомпозицията на домейн (Domain decomposition) [2] при паралелното програмиране се явява естествен подход при решаването на проблеми, при които за решаването на проблема за даден елемент  $D$  от домейна са нужни само малко подмножество от данни, които са "близо" до  $D$ . Накратко, идеята е, че разбиваме домейна на множество поддомейни и възлагаме решаването на проблема за всеки поддомейн на отделен процес. Важно е отделните поддомейни да са със еднаква големина за да може да се разпредели хубаво работата между процесорите. Съществуват два вида Domain decomposition - статичен и динамичен. При статичния в началото на алгоритъма разбиваме домейна и разпределяме работата между процесите. По време на симулацията разбиването не се променя, което може да води до намаляване на производителността тъй като данните при доста проблеми прескачат от един домейн в друг. Този проблем се решава от втория тип Domain decomposition, който по време на симулацията произчислява разбиването на домейна с цел балансиране на големината на отделните поддомейни. Domain decomposition намира приложение при решаването на проблеми от тип Cellular automata [3].

Тъй като и Wa-Tor попада в този тип проблеми, решението което представям тук е базирано на статична декомпозиция на домейна.

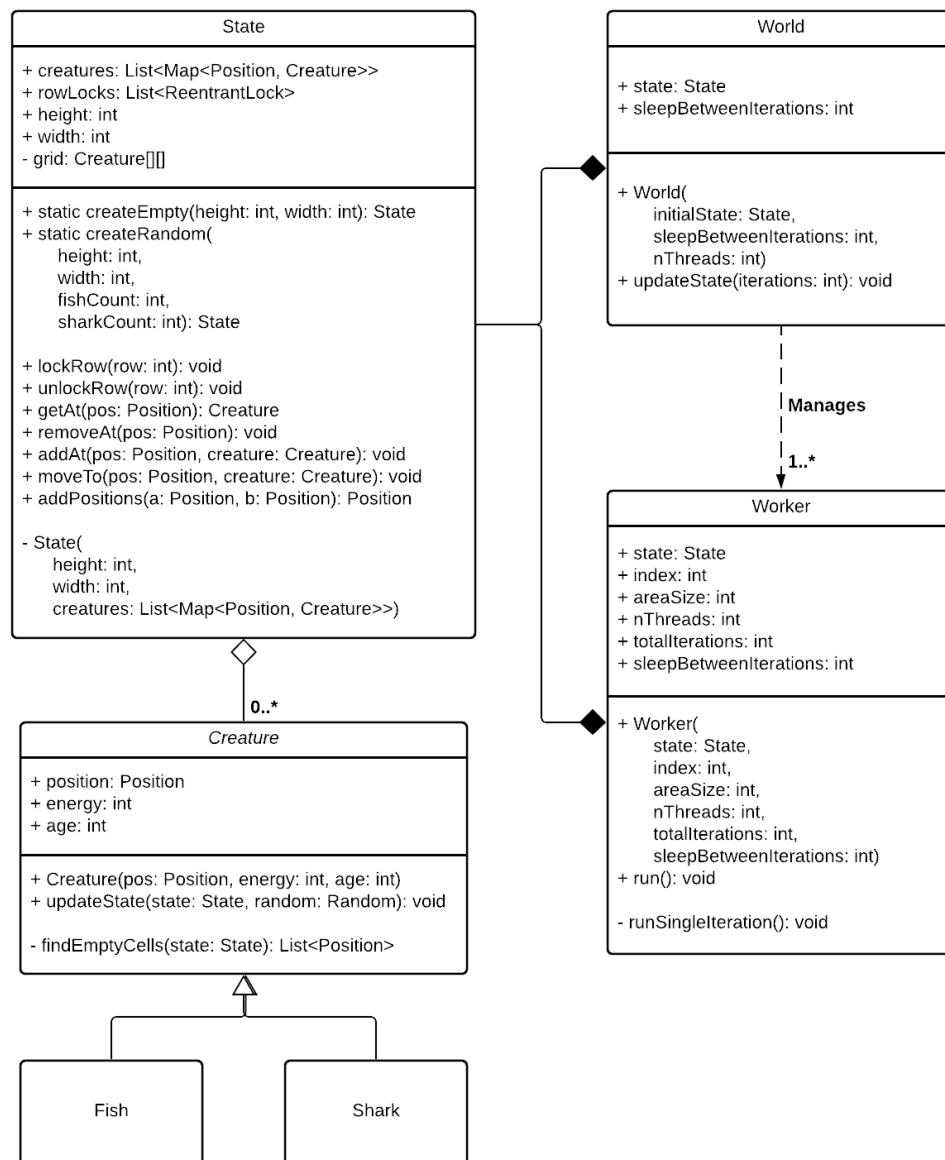
разбиваме светът по редове (или по колонии) на равни по големин ленти.

## 2 Архитектура

Решението е имплементирано на Java 16 и използва вградени способности за паралелно програмиране, част от Java SE.

- `java.lang.Thread`: основният начин за създаване на нишки във Java
- `java.util.concurrent.ExecutorService`
- `java.util.concurrent.locks.ReentrantLock`

Архитектурата използва статична декомпозиция на домейна и е по модела Master-Slaves.



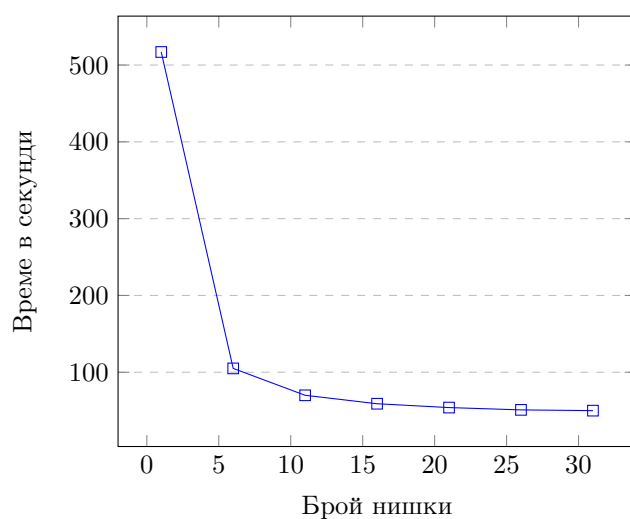
Фигура 1: UML Клас диаграма на проекта

### 3 Тестови резултати

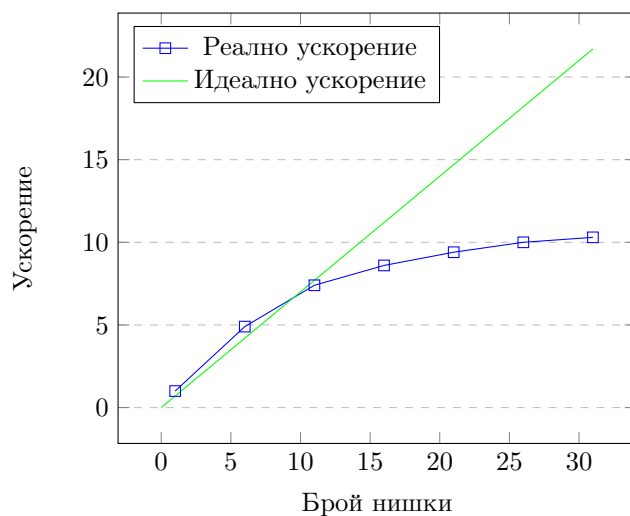
Характеристики на тестовата машина:

Ускорение (MIN,AVG,MAX) при размер 4000x2000 и популяция 1 000 000			
Брой нишки	MIN	AVG	MAX
1	1.00	1.00	1.00
6	4.78	4.89	4.93
11	7.08	7.38	7.50
16	8.44	8.63	8.65
21	9.24	9.41	9.36
26	9.61	10.00	10.04
31	10.12	10.32	10.39

Време за изгълнение при размер на полето 4000x2000

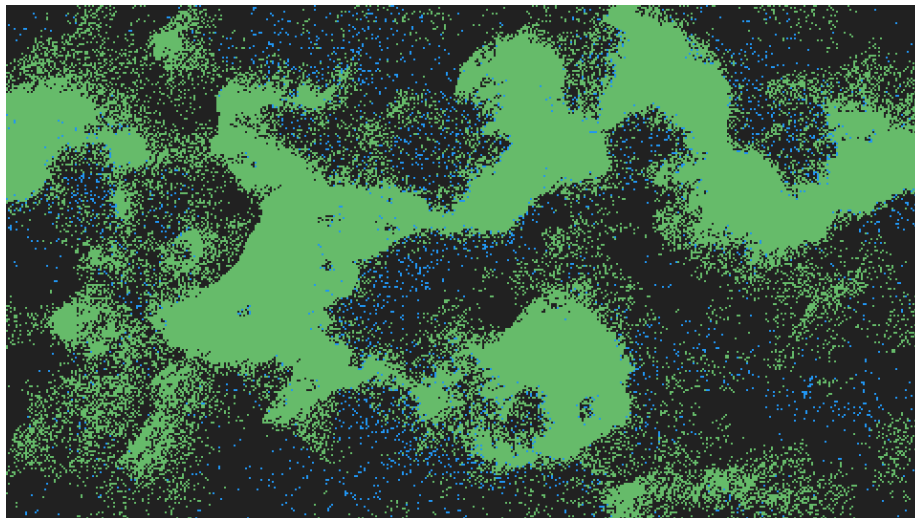


Средно ускорение при размер на полето 4000x2000

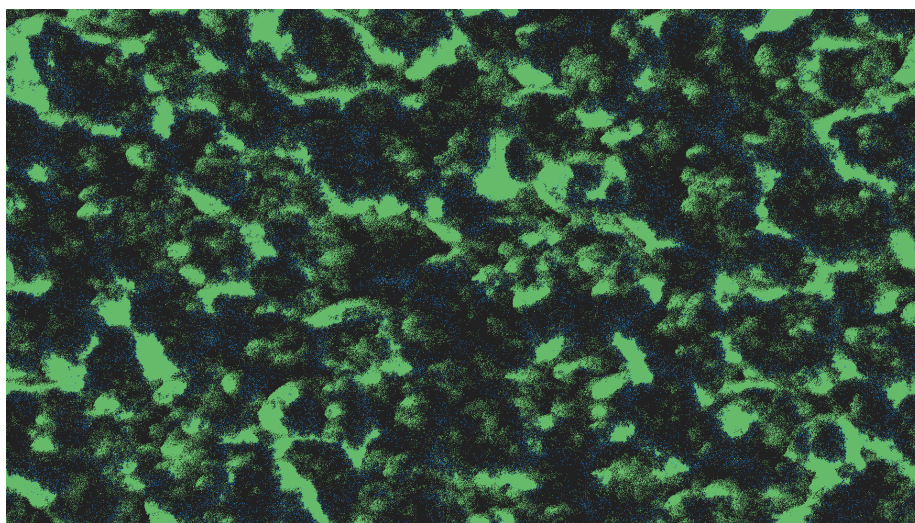


## 4 Визуализации

Освен симулацията, която служи за измерване на скалируемостта, в проекта е разработена и визуализация в реално време с помощта на Java Swing и AWT. Следват няколко екранни снимки от визуализации (херингите са в зелено, а акулите в синьо):



Фигура 2: Размер на полето 480x270, 10 000 херинги и 1000 хиляди акули



Фигура 3: Размер на полето 1920x1080, 1 000 000 херинги и 100 000 акули

Първоначалната цел на този "режим на работа" беше лесен начин за дебъгване, но в крайна сметка се получи и доста готина анимация, която би могла да се ползва за screensaver :).

## 5 Бъдещо развитие на проекта

Използване на Dynamic domain decomposition

## Литература

- [1] Alexander Keewatin Dewdney. Sharks and fish wage an ecological war on the toroidal planet wa-tor. *Scientific American*. pp. 14–22, 1984.
- [2] William Gropp. Parallel computing and domain decomposition. *Domain Decomposition Methods for Partial Differential Equations* (pp. 349-361), 1992.
- [3] Peter Slood and D.Talia. Cellular automata: promise and prospects in computational science. *Future Generation Computing Systems*, 1999.