УДК 681.3

С.М. Защипас, аспір.
Житомирський інженерно-технологічний інститут

ОБЧИСЛЕННЯ ЧАСУ ВІЗУАЛІЗАЦІЇ ТРИВИМІРНОЇ СЦЕНИ У ЗАДАЧІ СКЛАДАННЯ РОЗКЛАДУ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

(Представлено д.т.н., проф. Панішевим А.В.)

В статті розглядається метод обчислення часу візуалізації тривимірних сцен для застосування у задачі складання розкладу паралельних обчислень покадрової візуалізації в кластерних системах. Описується обчислювальний експеримент, на результатах якого було побудовано метод.

Запропоновано практичні рекомендації до застосування методу.

В імітаційному моделюванні, кінематографі, хімії, біології та інших галузях широко використовують покадрову візуалізацію тривимірних сцен.

Суть процесу візуалізації полягає в обчисленні кольору та інтенсивності кожної точки зображення відповідно до проходження світлового променя від джерел світла до спостерігача, з урахуванням різних параметрів видимих об'єктів: коефіцієнтів поглинання, відбиття, заломлення та інших. Крім того, зображення може бути при візуалізації оброблене спеціальними фільтрами, що надають зображенню якихось певних властивостей.

Процес візуалізації складних сцен, на яких міститься велике число елементів, може вимагати досить багато обчислювальних ресурсів. Наприклад, зображення, наведене на рис. 1, при візуалізації з чіткістю 1024 x 768 точок обчислюється на комп'ютері з процесором Intel Celeron-850 близько 50 хвилин.

Рис. 1. Приклад візуалізації тривимірного зображення

Досить часто для візуалізації тривимірних сцен використовують обчислювальні кластери [1], в яких процес візуалізації розпаралелюється. Кожен з вузлів кластера обробляє свою частину зображення або, у випадку, коли йдеться про деяку послідовність сцен, свій набір сцен. Програмне забезпечення для таких кластерів пишеться, звичайно, з використанням стандартних бібліотек розпаралелювання, таких, як MPI та PVM, робота яких базується на обміні повідомленнями між вузлами кластера [2, 3].

Для зменшення часу візуалізації та підвищення ефективності паралельних обчислень можна використовувати різні оптимізаційні методи, що базуються на теорії розкладів. Такі методи були запропоновані в [4, 5].

Розглянемо задачу про покадрову візуалізацію послідовності тривимірних сцен.

Нехай є послідовність A, що складається з n тривимірних сцен, заданих наборами властивостей їх об'єктів. Кожна з цих тривимірних сцен візуалізується за час $\tau_i, i \in [1; n]$. Для швидкого розв'язання таких задач звичайно використовують кластерні системи. В кластері послідовність A ділиться на m частин A_j, де $j \in [1; m]$, $A = A_1 \cup A_2 \cup ... \cup A_m$, $A_i \cap A_j = \varnothing, i \neq j$. Кожна із цих частин вирішується на своєму обчислювальному вузлі.

Вважаємо, що час τ_i є функцією деяких визначених параметрів тривимірної сцени, таких, як кількість елементів у видимій області, наявність "ефектів атмосфери", освітлення, відбивання та ін. На основі цього можна зробити висновок про, що τ_i досить суттєво відрізняються один від одного.

Задачу оптимального виконання візуалізації тривимірних сцен (кадрів) можна сформулювати в термінах класичної задачі теорії розкладів – розклад для мультипроцесорної системи (РМС).

РМС: задано кінцеву множину A кадрів, довжини візуалізації $l(a) \in Z^+$ для всіх $a \in A$, число $m \in Z^+$ та директивний термін $D \in Z^+$.

Необхідно знайти розбиття $A = A_1 \cup A_2 \cup ... \cup A_m$ множини A на m непересічних підмножин так, щоб

$$max\left(\sum_{a \in A_i} l(a); 1 \le i \le m \right) \le D . \quad (1)$$

Дослідженню даної задачі присвячено багато робіт. Найчастіше для рішення задачі застосовують або оптимальні алгоритми типу "гілок та меж", або "списочні" алгоритми, які дають наближене рішення за поліноміальний час.

Для складання розкладу покадрової візуалізації у кластерній системі необхідно знати час τ_i візуалізації. Як було сказано в постановці задачі, τ_i є функцією від великої кількості параметрів сцени, причому тривіального шляху визначення його не існує.

Припустимо, що час τ_i має деяку функціональну залежність

$$\tau_i = F(n) , \quad (2)$$

де n – кількість пікселів у зображенні.

Нехай

$$\overline{\tau_i} = \overline{F}(n) , \quad (3)$$

де $\overline{\tau_i}$ – середній час візуалізації одного піксела, а

$$\overline{F}(n) = \frac{F(n)}{n} . \quad (4)$$

З метою знаходження залежності $\overline{F}(n)$ було проведено такий обчислювальний експеримент: для декількох сцен вимірювався час візуалізації при різних значеннях n. За результатом обчислювального експерименту було побудовано графіки залежності $\overline{F}(n)$.

В експерименті використовувалося: комп'ютер з процесором Celeron-300 з 65 Мб пам'яті та вінчестером 6Гб, комп'ютер з процесором Celeron-850 з 128 Мб пам'яті та вінчестером 40 Гб. На кожному комп'ютері було встановлено операційну систему Linux (Дистрибутив ASP 7.1), та вільно розповсюджувану систему візуалізації тривимірних сцен PovRay [6].

Експеримент проводився з використанням стандартних демонстраційних тривимірних сцен, що розповсюджуються разом з PovRay: fish13.pov, chess2.pov, ionic5.pov. В цих файлах зберігається опис параметрів тривимірних сцен на С-подібній мові програмування.

Розміри картинки формувалися за формулою:

$$x = 2y , \; y \in [60;780], \; step 10 . \quad (5)$$

Кількість пікселів зображення розраховувалася за формулою:

$$n = xy . \quad (6)$$

На рис. 2 зображено графік функції $\overline{F}(n)$, який було побудовано за результатами візуалізації сцени fish13.pov на комп'ютері Celeron-800. З графіка видно, що на певному проміжку функція $\overline{F}(n) = \text{const}$.

Позначимо верхню межу кількості пікселів, при яких функція $\overline{F}(n)$ є нелінійною, як n^0.

Назвемо візуалізацію сцени у зображення із меншою кількістю пікселів порівняно з заданою *тестовою візуалізацією* сцени.

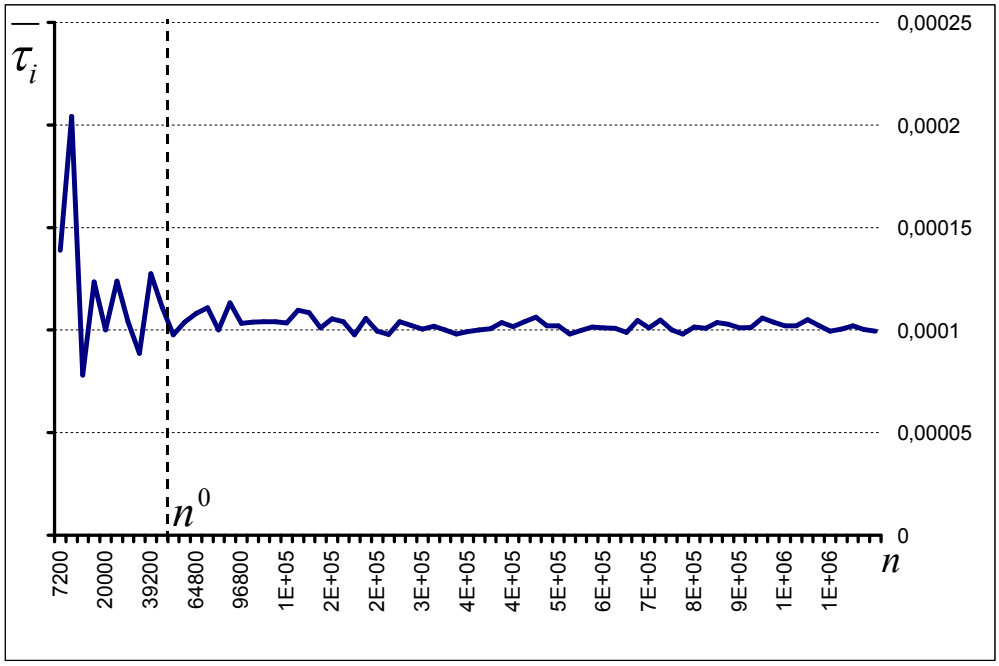

Рис. 2. Графік функції $\overline{F}(n)$ *візуалізації сцени* fish13.pov

З графіка видно, що на проміжку $\left[n^0;10^6\right]$, при $n^0 \approx 50000$ функція $\overline{F}(n) \approx 0{,}0001$. Середнє значення функції на цьому проміжку є $\overline{\overline{F}(n)} \approx 0{,}0001024$. Мінімальне та максимальне значення функції на цьому проміжку відповідно є $\overline{F}(n)_{\min} \approx 0{,}0000977$ та $\overline{F}(n)_{\max} \approx 0{,}0001134$, що дає максимальне відхилення в 11 % від середнього значення.

А на проміжку $\left[7200;\, n^0\right)$ функція є нелінійною та має вигляд ламаної.

Наступний графік (рис. 3) побудовано за результатами візуалізації сцени chess2.pov на комп'ютері Celeron-850. Як видно з графіка, на проміжку $\left[n^0;10^6\right]$, при $n^0 \approx 50000$ функція $\overline{F}(n) \approx 0{,}0037$.

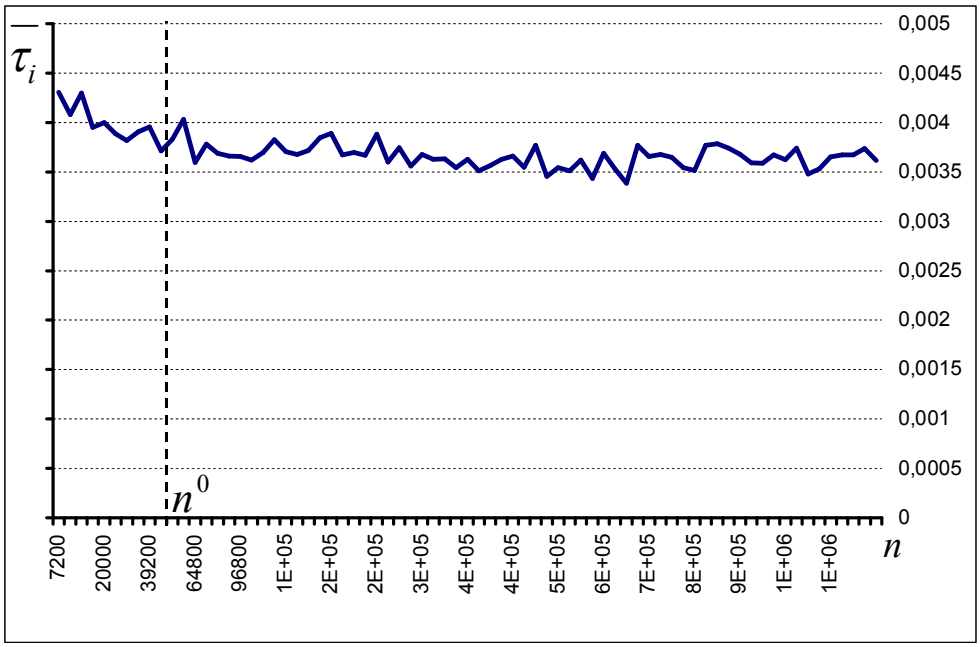

Рис. 3. Графік функції $\overline{F}(n)$ *візуалізації сцени chess2.pov*

Середнє значення функції на цьому проміжку є $\overline{\overline{F}(n)} \approx 0{,}003656$. Мінімальне та максимальне значення функції на цьому проміжку відповідно є $\overline{F}(n)_{\min} \approx 0{,}00338$ та $\overline{F}(n)_{\max} \approx 0{,}00403$, що дає

максимальне відхилення в 10 % від середнього значення. А на проміжку $\left[7200; n^0\right)$ функція має тенденцію до зростання. Подібну поведінку функції можна пояснити збільшенням витрат на ініціалізацію алгоритму та випадковістю процесів у багатопотоковому середовищі операційної системи Linux. Крім того, при проведенні цього експерименту, який тривав протягом 32 годин, ймовірність випадкових процесів була набагато вищою в порівнянні з візуалізацією сцени fish13.pov, час експерименту з якою зайняв всього 1,6 год.

На рис. 4 зображено графіки функції $\overline{F(n)}$, побудовані за результатами візуалізації сцени ionic5.pov на комп'ютері Celeron-850 та на комп'ютері Celeron-300.

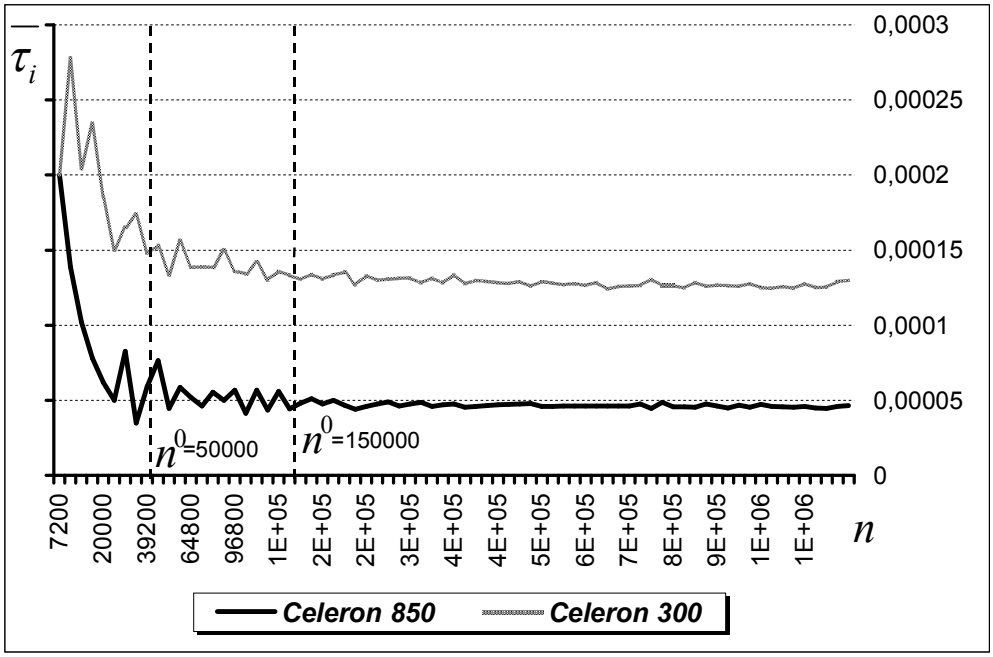

Рис. 4. Графіки функції $\overline{F(n)}$ *візуалізації сцени* ionic5.pov

Були отримані наступні результати.

Для комп'ютера Celeron- 850:

середнє значення функції на проміжку $\left[n^0;10^6\right]$, при $n^0 \approx 50000$ є $\overline{\overline{F(n)}} \approx 0{,}0000473$. Мінімальне та максимальне значення функції на цьому проміжку відповідно є $\overline{F(n)}_{\min} \approx 0{,}0000413$та $\overline{F(n)}_{\max} \approx 0{,}0000586$, що дає максимальне відхилення в 23 % від середнього значення.

Середнє значення функції на проміжку $\left[n^0;10^6\right]$, при $n^0 \approx 150000$ є $\overline{\overline{F(n)}} \approx 0{,}0000465$. Мінімальне та максимальне значення функції на цьому проміжку відповідно є $\overline{F(n)}_{\min} \approx 0{,}0000439$та $\overline{F(n)}_{\max} \approx 0{,}0000586$, що дає максимальне відхилення в 9,6 % від середнього значення.

Для комп'ютера Celeron- 300:

середнє значення функції на проміжку $\left[n^0;10^6\right]$, при $n^0 \approx 50000$ є $\overline{\overline{F(n)}} \approx 0{,}00013$. Мінімальне та максимальне значення функції на цьому проміжку відповідно є $\overline{F(n)}_{\min} \approx 0{,}000124$та $\overline{F(n)}_{\max} \approx 0{,}000156$, що дає максимальне відхилення в 20 % від середнього значення.

Середнє значення функції на проміжку $\left[n^0;10^6\right]$, при $n^0 \approx 150000$ є $\overline{\overline{F(n)}} \approx 0{,}0001281$. Мінімальне та максимальне значення функції на цьому проміжку відповідно є $\overline{F(n)}_{\min} \approx 0{,}000124$та $\overline{F(n)}_{\max} \approx 0{,}000135$, що дає максимальне відхилення в 5,57 % від середнього значення.

З графіків рис. 2–4 видно, що при деякому $n \in [n^0; n_i]$, де n_i – кількість пікселів i -ї сцени, функція $\overline{F}(n) \approx k_i$, $k_i = \text{const}$, де k_i – середній час розрахунку одного піксела i -ї сцени. Виходячи з формули (4)

$$\frac{F(n)}{n} = k_i, n \in [n^0; n_i] \quad (7)$$

або

$$F(n) = nk_i, n \in [n^0; n_i]. \quad (8)$$

З формули (8), що при $k_i = \text{const}$, $F(n)$ – лінійна функція аргументу *n*.

Виходячи з формули (2), можна записати

$$\tau_i^0 = n^0 k_i, \quad (9)$$

де τ_i^0 – час візуалізації *i*-ї сцени при $n = n^0$,

$$\tau_i = n_i\ k_i. \quad (10)$$

Склавши пропорцію, отримуємо

$$\tau_i = \frac{n_i}{n^0}\tau_i^0. \quad (11)$$

Формула (11) виражає час τ_i візуалізації n_i пікселів i-ї сцени через час τ_i^0 візуалізації n^0 пікселів цієї сцени. У даному випадку візуалізація n^0 пікселів сцени буде тестовою.

Для практичного використання формули (11) необхідно визначити n^0 та τ_i^0 для відповідної сцени.

З графіка (рис. 4) видно, що вигляд функції $\overline{F}(n)$ не змінюється. Відбувається лише її зміщення відносно осі $\overline{\tau_i}$, відповідно до продуктивності обчислювальної системи. Причому змінюється максимальне відхилення від середнього значення $\overline{\overline{F}(n)}$.

Максимальне відхилення від середнього значення дещо залежить від параметрів сцени. Це говорить про те, що n^0 також залежить від цих параметрів. Якщо порівняти графіки рис. 2 та рис. 4, то видно, що функція $\overline{F}(n)$ для комп'ютера Celeron-850 є нелінійною на відрізку, для якого $\tau_i < t^0$, $t^0 \approx 8c$, t^0 – деяка константа, що не залежить від параметрів сцени та характеризує випадкові процеси, затрати на ініціалізацію алгоритму та похибки вимірювання часу на малих інтервалах. Якщо порівняти графіки рис. 3 та рис. 4, то видно, що час візуалізації одного піксела відрізняється майже в тисячу разів, а відхилення від середнього значення є 10 % для chess2.pov (рис. 3) та 23 % для ionic5.pov (рис. 4). Це дає підставу стверджувати, що n^0 залежить від продуктивності обчислювальної системи. Залежністю n^0 від параметрів сцени можна знехтувати, якщо час тестової візуалізації $\tau_i^0 > t^0$. Тобто для кожної обчислювальної системи можна знайти індивідуальне значення n^0, що буде задовольняти всі сцени, для яких $\tau_i^0 > t^0$.

Параметри t^0 та n^0 для обчислювальної системи підбираються індивідуально за допомогою послідовності тестових візуалізацій деякого набору сцен. Параметр τ_i^0 знаходиться тестовою візуалізацією сцени при $n = n^0$.

Отже, на основі результатів експерименту можна зробити висновок, що існує можливість визначати час τ_i для подальшого використання цього часу у складанні розкладу паралельної візуалізації послідовності тривимірних сцен.

Очевидно, що ефективність використання розкладу у паралельній системі візуалізації залежить від терміну визначення часу візуалізації сцени.

Виходячи з (11), використання цього методу буде ефективним за умови:

$$n^o << n_i. \quad (12)$$

Використання методу визначення часу візуалізації для складання розкладу паралельних обчислень в кластері повинно проходити за такою схемою:

1. Ведучий вузол кластера не повинен виконувати ніяких задач, крім даної.
2. Для ведучого вузла кластера обчислюються t^0 та n^0 за допомогою послідовності тестових візуалізацій деякого набору сцен.
3. Послідовність тривимірних сцен перевіряється на умову (12), якщо вона не справджується, то відбувається паралельна візуалізація без складання розкладу.
4. Для кожної сцени проводиться тестова візуалізація з $n = n^0$ та виміряється час τ_i^0.

5. Оскільки для складання розкладу можна використовувати відносний покажчик часу, то в підсистему складання розкладу передається набір τ_i^0 .
6. Підсистема складання розкладу повертає розбиття списку сцен на підсписки.
7. Виконується паралельна візуалізація відповідно до розбиття.
8. Виконується збір та сортування отриманих зображень.

Подібний алгоритм може бути використаний як до послідовності тривимірних сцен, так і до розбитої на частини однієї тривимірної сцени.

ЛІТЕРАТУРА:

1. *Андреев А., Воеводин Вл., Жуматий С.* Кластеры и суперкомпьютеры – близнецы или братья? – http://www.osp.ru/os/2000/05-06/009.htm
2. MPI:A Message-Passing Interface Standard –http://parallel.ru/docs/Parallel/mpi1.1/mpi-report.html
3. PVM: Parallel Virtual Machine – http://www.epm.ornl.gov/pvm/
4. Теория расписаний и вычислительных машин / Под ред. Э.Г. Кофмана. – Москва: Наука, 1984.
5. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. – М.: Мир, 1982 – 416 с.
6. PovRay – http://www.povray.org

ЗАЩИПАС Сергій Миколайович – аспірант Житомирського інженерно-технологічного інституту.
Наукові інтереси:
– теорія розкладів;
– паралельні обчислення;
– операційні системи.

Подано 17.03.2002

Защипас С.М. Обчислення часу візуалізації тривимірної сцени у задачі складання розкладу паралельних обчислень

Защипас С.Н. Вычисление времени визуализации трехмерной сцены в задаче составления расписания параллельных вичислений.

Zaschipas S.M. Computing of visualization time of 3D scene in task of schedule creation for parallel processing.

УДК 681.3

Ввычисление времени визуализации трехмерной сцены в задаче составления расписания параллельных вичислений / С.Н. Защипас

В статье рассматривается метод вычисления времени визуализации трехмерных сцен для применения в задаче составления расписания параллельных вычислений покадровой визуализации в кластерных системах. Описывается вычислительный эксперимент, на результатах которого был построен метод.

Предложены практические рекомендации по применению метода.

УДК 681.3

Computing of visualization time of 3D scene in task of schedule creation for parallel processing / S.M. Zaschipas.

The article elucidates the method of computing of 3d visualization time in for use in task of schedule creation for parallel processing in cluster systems. Describes computing experiment which results are method basis.

Practical recommendations of method using are suggested.