# Модель структурированных ячеек для параллельных вычислений с распределенной памятью

П.А. Васёв[1], С. В. Поршнев[2]

ИММ УрО РАН им Н. Н. Красовского[1],
Уральский федеральный университет им. Б. Н. Ельцина[2]

В работе предлагается новая модель параллельного программирования. Она вдохновлена моделью Entity-Component-System (ECS), которая применяется при разработке игр и систем моделирования на машинах с общей памятью [1]. Модель ECS обладает свойствами вычислительной эффективности и ясности при программировании. Важно, что эта модель опирается на строгий порядок выполнения систем (system). Этот порядок невозможно поддержать в системах с распределенной памятью без специальных доработок, чему и посвящена настоящая работа. Вводится следующая новая модель вычислений.

Понятия модели.

Мир (world) — область, содержащая сущности, правила и ссылки.

Сущность (entity) — это единица данных. Сущность имеет глобальный идентификатор в рамках вычисления. К сущности прикрепляются компоненты. Сущность с компонентами есть структурированная ячейка.

Компонента (component) — это элемент данных, прикрепленный к конкретной сущности. Компонент имеет локальный идентификатор в рамках сущности (другими словами имя).

Правило (rule, system) — предписание о том, что при наличии у сущности заданного списка компонент (список имен компонент), следует выполнять функцию преобразования над этими компонентами, а результат записать в эту же сущность в результирующие компоненты.

Ссылка (binding) — предписание о том, что при записи компоненты заданной сущности следует произвести копирование этой компоненты в заданную компоненту другой сущности.

Рабочий — процесс, который обеспечивает работу системы. Подразумевается, что рабочие являются однопоточными процессами операционной системы, запускаются на узлах суперкомпьютера, в количестве согласно числу процессоров или видеокарт на узле.

Операции модели.

Создать сущность — создает сущность на указанном рабочем в указанном мире и возвращает ее глобальный идентификатор.

Обновить компоненту — обновляет заданную компоненту указанной сущности.

Добавить правило — добавляет правило в мир.

Добавить ссылку — добавляет ссылку в мир.

Описание параллельной программы.

Предполагается, что программа запустит необходимое число рабочих на выделенных задаче узлах суперкомпьютера. Затем создаст необходимые сущности в памяти этих рабочих. Подразумевается, что эти сущности выражают декомпозицию вычислительных данных.

Далее программой выражается вычислительный алгоритм. Для этого добавляются ссылки для описания передачи обновления компонент между сущностями, соседними в смысле декомпозиции данных, чтобы обеспечить обмен граничными значениями. И добавляются правила, которые описывают порядок преобразования компонент сущностей.

Вычислительная модель работает следующим образом. Каждый рабочий независимо и параллельно от остальных рабочих анализирует свои локальные сущности и правила, и находит пары (сущность, правило), готовые к исполнению. Далее он исполняет их. Это приводит к обновлению состояния локальных сущностей и также удаленных сущностей в памяти других рабочих, исходя из введенных ранее ссылок.

Сущность готова к применению правила, если:

1. В сущности присутствуют все входные компоненты, перечисленные в правиле.
2. Эти входные компоненты имеют одинаковый счетчик обновления.
3. Все ссылки из выходных компонент, которые будет записывать данное правило, не заблокированы.

Пункты 2 и 3 нуждаются в особом пояснении. Особенность модели ECS заключается в последовательной обработке сущностей правилами (системами в терминологии ECS). Однако в распределенной среде это невозможно гарантировать без применения особых мер.

Для компонент вводится счетчик обновлений. Считается, что компоненты должны обновляться на каждой итерации моделирования. Вводится требование одинакового значения счетчика обновлений всех компонент для запуска правила. Это позволяет гарантировать, что правила выполняются в заданной последовательности в контексте других правил.

Блокировка исходящих ссылок работает по следующему алгоритму. Каждая компонента возможно связана с другими компонентами других сущностей ссылками. Однако при обновлении компоненты нельзя сразу передавать данные согласно ссылке, потому что старая копия целевой компоненты может быть еще не обработана всеми подходящими правилами текущей итерации. Поэтому пересылка производится в два этапа: 1) запрос на готовность обновления удаленной компоненты, и после получения ответа о готовности 2) обновление этой компоненты и счетчика ее обновлений.

Описанный механизм обеспечивает правильный (заданный программистом) детерминированный порядок обработки данных, что и требуется для модели ECS.

Модель находится в стадии разработки. 1) решается вопрос миграции вычислений. В текущей версии все вычисления привязаны к сущностям. Это позволяет организовать параллельное вычисление, и также позволяет осуществлять миграцию вычислений, но сразу всех над данной сущностью. 2) В модели нет возможности расщепить стадии вычисления над сущностью на отдельные узлы суперкомпьютера. В промышленных системах эта проблема решена путем явного разделения сущностей и передач данных между ними [2].

Авторы предполагают, что наиболее удобно и эффективно в предложенной модели этого можно добиться путем разделения вычислений между мирами и налаживания связей между ними. Это позволит использовать вычислительные ресурсы как общие между мирами (т.е. между разными явлениями), так и распределять их. Решение находится в стадии проработки.

На рис. 1. показан пример работы прототипа предложенной модели. Вычислительная программа выполняет на трехмерном воксельном кубе игру «жизнь», и одновременно проводится онлайн-визуализация этого вычисления. Использован язык Python, сокеты, asyncio.

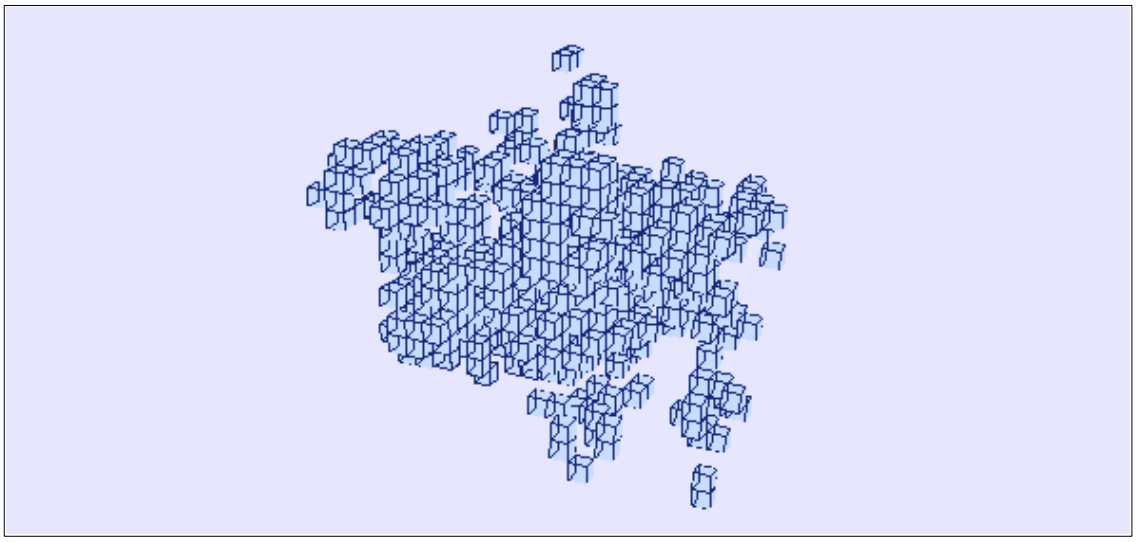

**Рисунок 1.** Онлайн-визуализация параллельного вычисления игры «жизнь»

## Литература

1. Entity Component System, URL: https://en.wikipedia.org/wiki/Entity_component_system.
2. Надуев А.Г., Черевань А.Д., Лебедева А.С. Архитектура программного модуля «ЛОГОС ПЛАТФОРМА» // Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов. 2022. № 4. С. 55–63. DOI: 10.53403/24140171_2022_4_55.