Системный анализ и проектирование сложных систем

Домашнее задание №1

ФИО, № группы: Ликсаков Максим Владимирович, МКС234

Возьмите систему массового обслуживания, с которой вы сталкиваетесь. Минимальные требования к системе/модели: 4 ресурса; использование/задействование 2-х параметров агентов и использование хотя бы одного из них в работе какого-либо ресурса; 3 условия/перехода; стохастические показатели работы элементов; завершение работы системы по числу обслуженных агентов.

1. Опишите работу системы. Выберите временной интервал моделирования. Укажите упрощения, допущения, которые вы используете при создании модели системы. Укажите потенциальную цель/назначение анализа этой системы. Постройте схему системы (1 балл).
2. Разработайте имитационную модель системы в программе AnyLogic (3 балла).
3. Создайте 2D и 3D анимацию модели (1 балл).
4. Осуществите сбор статистики работы модели, добавьте минимум 1 диаграмму для ресурса, 1 диаграмму для очереди, 1 временной график, 1 гистограмму, 1 блок измерения времени. Для каждого ресурса определите коэффициент загрузки, среднее время обработки, число транзактов/агентов получивших отказ в обслуживании. Для каждой очереди определите среднюю длину очереди, среднее время пребывания требований в очереди (1 балл).
5. Добавьте на экран эксперимента и запуска модели элемент управления интенсивностью поступления агентов. Увеличьте в 2 раза интенсивность поступления агентов. Опишите словами, как изменятся результаты моделирования работы системы. Уменьшите в 2 раза по сравнению с исходным значением интенсивность поступления требований. Опишите словами, как изменятся результаты моделирования работы системы (1 балл).
6. Проведите эксперимент по варьированию минимум 2-х параметров модели и визуализируйте результаты. При визуализации отобразите значения варьируемых параметров и значения минимум 2-х результатов работы системы (2 балла).
7. Выберите целевую функцию, которая связана с потенциальной целью/назначением анализа системы. Проведите оптимизационный эксперимент. Опишите полученный результат (1 балл).

1. Опишите работу системы. Выберите временной интервал моделирования. Укажите упрощения, допущения, которые вы используете при создании модели системы. Укажите потенциальную цель/назначение анализа этой системы. Постройте схему системы (1 балл).

Описание Системы

Apache Spark — это быстрая и обобщенная система вычислений для больших данных, предоставляющая высокоуровневые API на Java, Scala, Python и R, а также предоставляющая оптимизированный движок, который поддерживает общие вычислительные графы. Он также поддерживает набор высокоуровневых инструментов, включая Spark SQL для SQL и структурированных данных, MLlib для машинного обучения, GraphX для обработки графов и Structured Streaming для обработки потоков.

Основные Компоненты и Принципы Работы

- **YARN**: YARN (Yet Another Resource Negotiator) используется для управления ресурсами и распределения задач по вычислительным узлам.
- **Computing Nodes**: Вычислительные узлы обрабатывают задачи, распределенные YARN, и взаимодействуют друг с другом для обмена данными и результатами.
- **User Tasks**: Задачи пользователей имеют различные атрибуты и параметрыю

Цель Анализа

Анализ работы Apache Spark может помочь в оптимизации производительности, управлении ресурсами и улучшении обработки больших данных, что, в свою очередь, может привести к более эффективному использованию ресурсов и улучшению качества обслуживания пользователей.

Упрощения и Допущения

При создании модели были сделаны некоторые упрощения и допущения для упрощения процесса моделирования, такие как равномерное распределение задач и использование двух типов узлов с различными характеристиками. Эти упрощения необходимо учитывать при анализе результатов моделирования.

Для более глубокого понимания работы Apache Spark рекомендуется изучить предоставленные выше источники и обучающие материалы.

Компоненты Модели Системы

- **user_task**:
  - **Описание**: Задачи от пользователей.
  - **Интенсивность**: 20 задач в минуту.
  - **Атрибуты**:
    - `query`: Сложность запроса.
    - `timeout`: Время ожидания, равномерно распределено между 40 и 60 секундами.
- **YARN**:
  - **Описание**: YARN (Yet Another Resource Negotiator) действует как менеджер ресурсов и распределяет задачи пользователя по доступным вычислительным узлам.
  - **Функции**:
    - **Распределение Задач**: YARN принимает задачи от пользователей и распределяет их по вычислительным узлам на основе доступных ресурсов и других факторов.
    - **Мониторинг Ресурсов**: YARN отслеживает использование ресурсов и доступность узлов для оптимизации распределения задач.

        - **Обработка Сбоев и Таймаутов**: YARN обрабатывает ситуации, когда задача "застряла" или превысила установленный таймаут, и принимает соответствующие меры.
    - **Выходы**: По таймауту и если задача "застряла", YARN может принять решение о прекращении или перезапуске задачи.
- **YARN Output**:
    - **Описание**: Выбирает узел для выполнения задачи.
    - **Распределение задач**: Равномерное (0.2 для каждого узла).
- **Computing Nodes**:
    - **Описание**: Пять вычислительных узлов, которые обрабатывают задачи, распределенные YARN. Узлы разделены на два типа в зависимости от их характеристик и возможностей.
    - **Тип 1**:
        - **Количество**: Три узла.
        - **Характеристики**: Узлы этого типа в среднем завершают задачу за 1 минуту.
        - **Ядра**: Каждый узел имеет 4 ядра, что позволяет выполнять 4 задачи параллельно.
        - **Действия**: Узлы этого типа принимают задачи от YARN, обрабатывают их, выполняя необходимые трансформации и вычисления, и затем отправляют результаты на агрегацию.
    - **Тип 2**:
        - **Количество**: Два узла.
        - **Характеристики**: Узлы этого типа в среднем завершают задачу за 2 минуты.
        - **Ядра**: Каждый узел имеет 8 ядер, что позволяет выполнять 8 задач параллельно.
        - **Действия**: Подобно узлам типа 1, узлы типа 2 принимают задачи, обрабатывают их, выполняя необходимые трансформации и вычисления, и отправляют результаты на агрегацию.
- **reduce_node**:
    - **Описание**: Собирает результаты выполнения задач с узлов и формирует итоговый результат.
    - **Время агрегации**: 30 секунд.
- **response**:
    - **Описание**: Выход системы. Отправляет результаты пользователю.

Сравнение Модели и Реальной Работы Apache Spark

Модель включает в себя несколько аспектов работы Apache Spark, однако существуют некоторые упрощения и различия. Ниже приведены некоторые точки сравнения между моделью и реальной работой Apache Spark:

1. Распределение Задач:

- **Модель:** Задачи равномерно распределяются по узлам.
- **Apache Spark:** Задачи распределяются на основе локальности данных и доступных ресурсов, а не равномерно.

2. Типы Узлов:

- **Модель:** Узлы разделены на два типа с различными временами обработки и возможностями параллельной обработки задач.
- **Apache Spark:** Обычно все узлы в кластере Spark однородны, но Spark может работать с разнородным оборудованием.

3. Выполнение Задач:

- **Модель:** Задачи выполняются узлами, а результаты агрегируются `reduce_node`.
- **Apache Spark:** Задачи выполняются по этапам, и каждый этап выполняет трансформацию или действие над данными. Результаты задач внутри этапа агрегируются с использованием функции reduce.

4. YARN:

- **Модель:** YARN представлен как очередь, управляющая распределением задач.
- **Apache Spark:** YARN — это менеджер кластера, который может использоваться Spark для выделения ресурсов, но Spark также может использовать другие менеджеры кластера, такие как Mesos или свой собственный менеджер кластера.

5. Атрибуты Задач Пользователя:

- **Модель:** Задачи пользователя имеют атрибуты, такие как `query` и `timeout`.
- **Apache Spark:** Задачи пользователя в Spark больше связаны с трансформациями и действиями над RDDs/DataFrames, и обычно у них нет атрибута «timeout».

6. Сбой Задач:

- **Модель:** Задачи могут завершиться неудачей, если они ждут больше, чем параметр таймаута.
- **Apache Spark:** Задачи могут завершиться неудачей по различным причинам, таким как сбой узла, и у Spark есть механизмы для обработки сбоев задач, такие как повторное выполнение задачи на другом узле.

7. Агрегация Задач:

- **Модель:** `reduce_node` агрегирует информацию от задач за 30 секунд.
- **Apache Spark:** Агрегация в Spark обычно выполняется с использованием трансформаций, таких как `reduce` или `aggregate`, и время, которое это занимает, зависит от объема данных и сложности функции агрегации.

Создание модели симуляции часто включает в себя создание упрощений и предположений для обеспечения управляемости модели, поэтому нормально, что модель не улавливает все детали реальной системы. Главное — убедиться, что модель точно представляет те аспекты системы, которые важны для ответов на вопросы исследования.

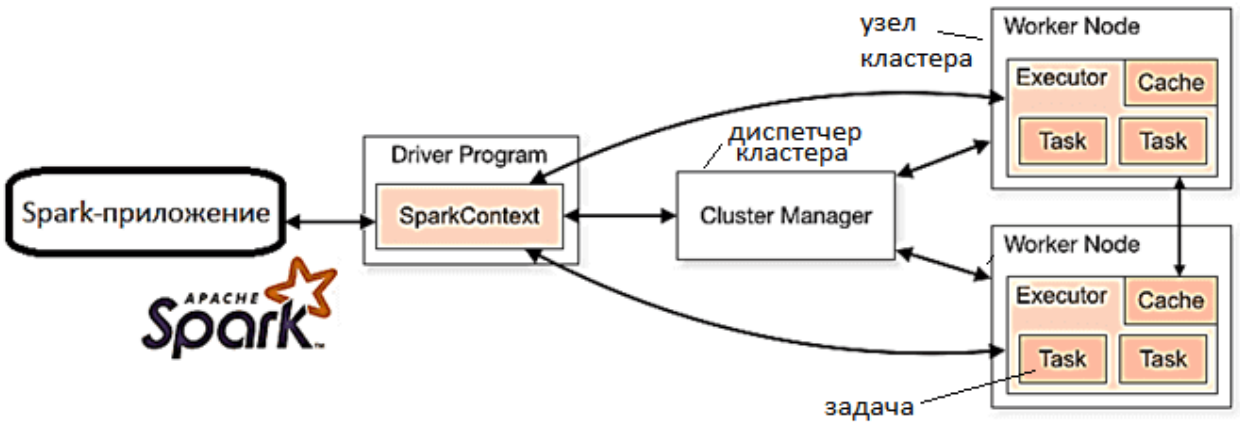


Рисунок 1 – Схема системы

2. Разработайте имитационную модель системы в программе AnyLogic (3 балла).

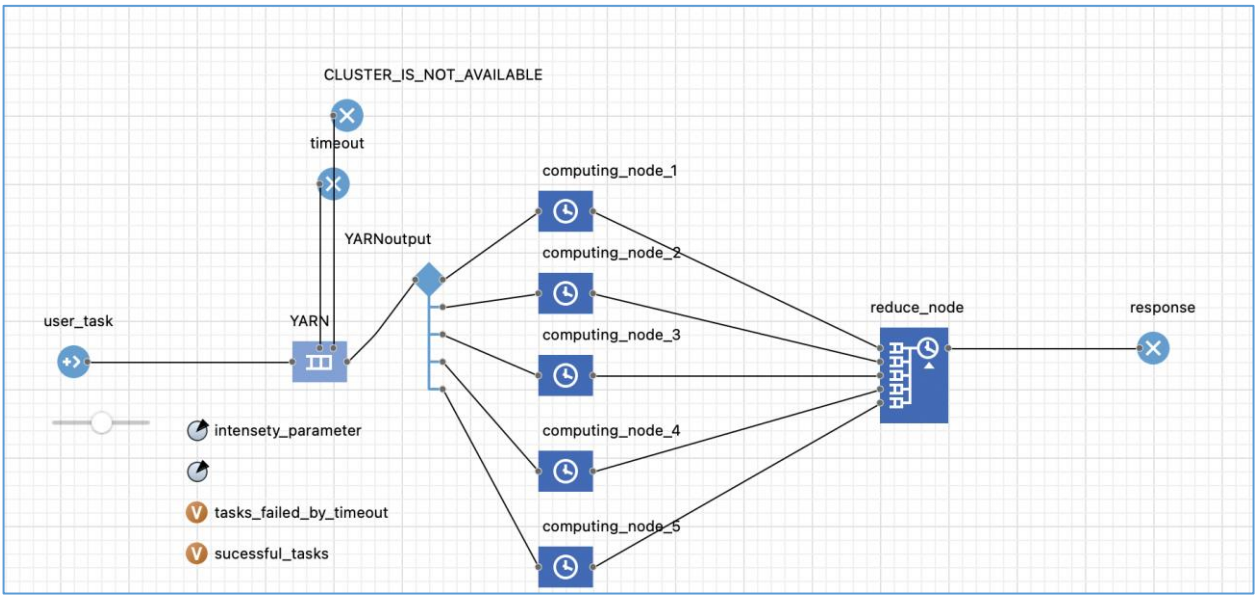


Рисунок 2 – Имитационная модель системы в AnyLogic

Элемент	Параметр/действие	Значение/код
user_task	**Интенсивность**	20 задач в минуту.
	query: Сложность запроса	
	timeout: Время ожидания	равномерно распределено между 40 и 60 секундами.
YARN	**Выход в Yarn Output**	
	Выход по таймауту	По значению timeout из user_task
	Выход по вытеснению	Вытеснение агентов
YARN Output	**Распределение задач**	равномерное (0.2 для каждого узла).
Computing Nodes	**Время работы узла**	2 минуты
	Ядра	4-8 (в зависимости от типа 1-2)
reduce_node	**Время агрегации задач**	30 секунд
response	**Выходящий поток ответов**	

Подробное описание всех компонентов есть в Главе 1. Ее содержимое было составлено до Шаблона ДЗ.

Таблица 1 – Настроенные параметры и действия свойств элементов модели

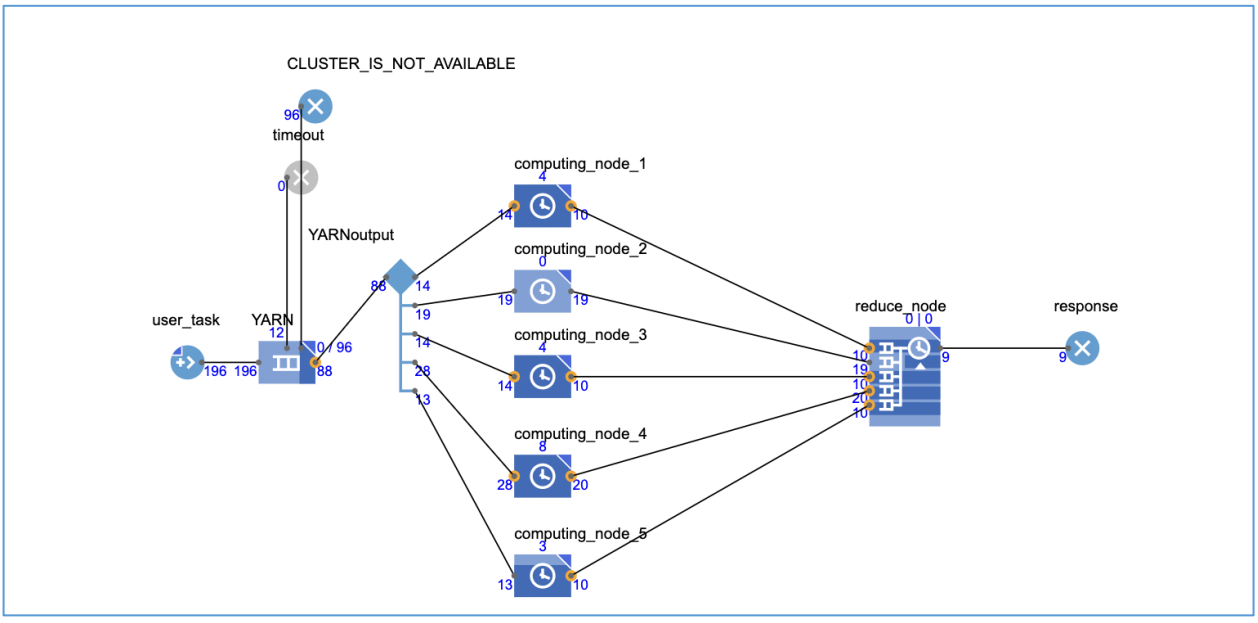


Рисунок 3 – Результаты работы (запуска) модели

3. Создайте 2D и 3D анимацию модели (1 балл).

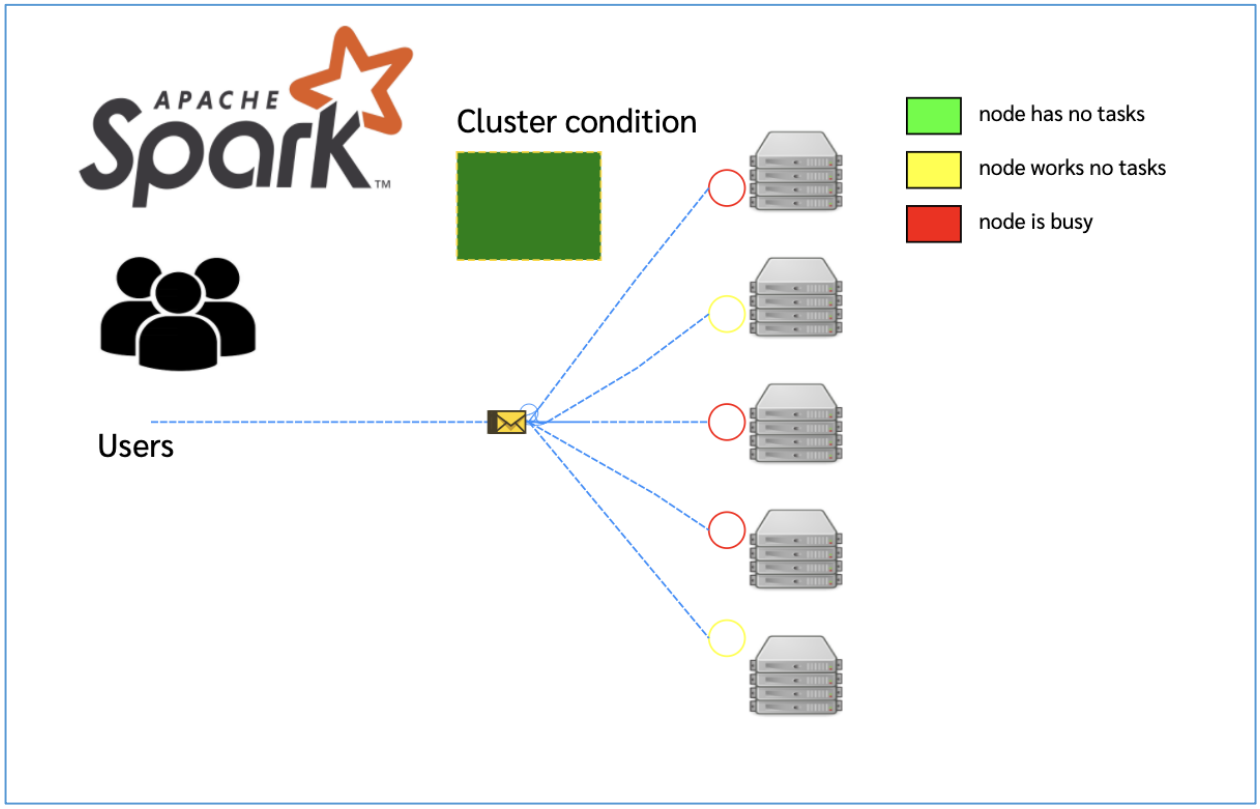


Рисунок 4 – 2D анимация

-

Рисунок 5 - 3D анимация

4. Осуществите сбор статистики работы модели, добавьте минимум 1 диаграмму для ресурса, 1 диаграмму для очереди, 1 временной график, 1 гистограмму, 1 блок измерения времени. Для каждого ресурса определите коэффициент загрузки, среднее время обработки, число транзактов/агентов получивших отказ в обслуживании. Для каждой очереди определите среднюю длину очереди, среднее время пребывания требований в очереди (1 балл).

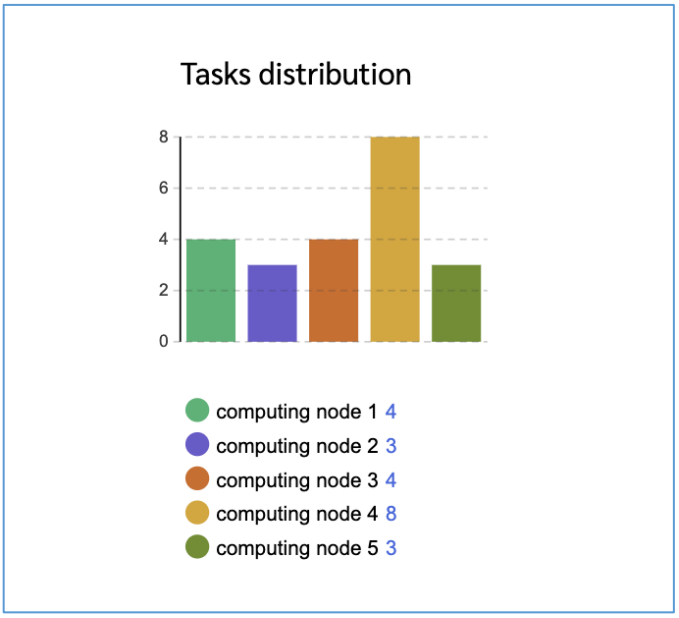


Рисунок 6 – Диаграммы для ресурсов

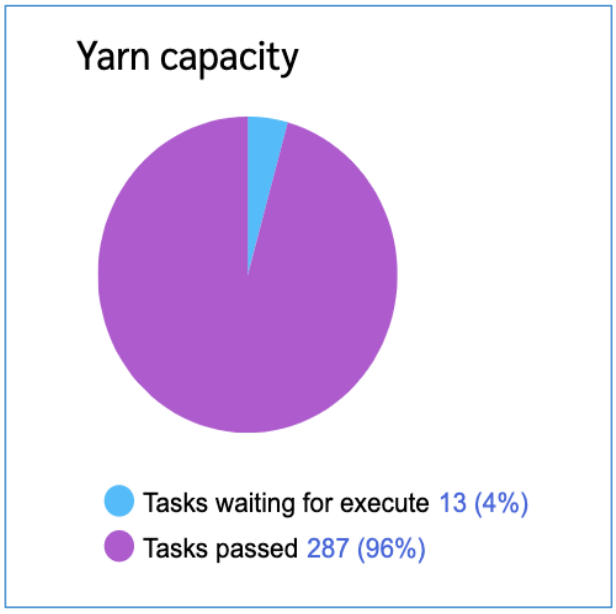


Рисунок 7 – Диаграммы для очередей

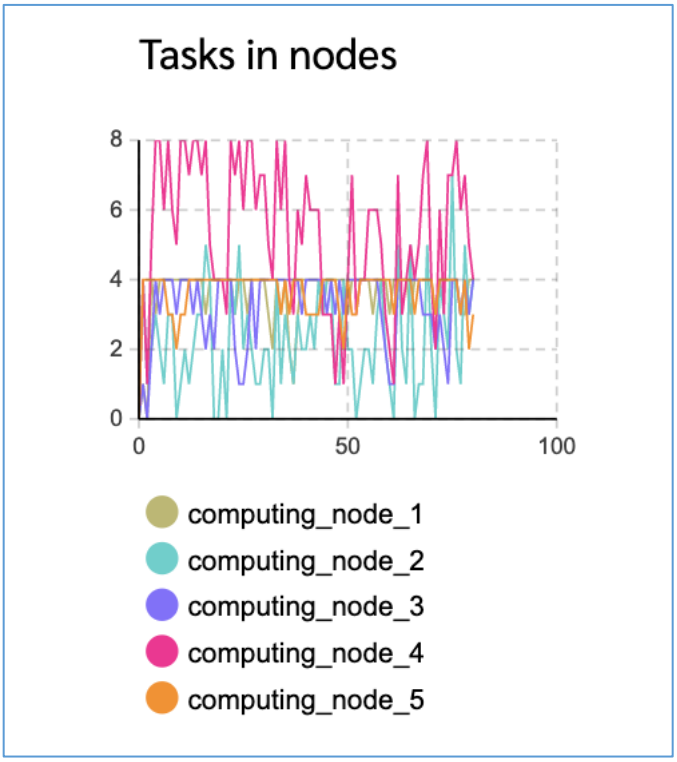


Рисунок 8 – Графики

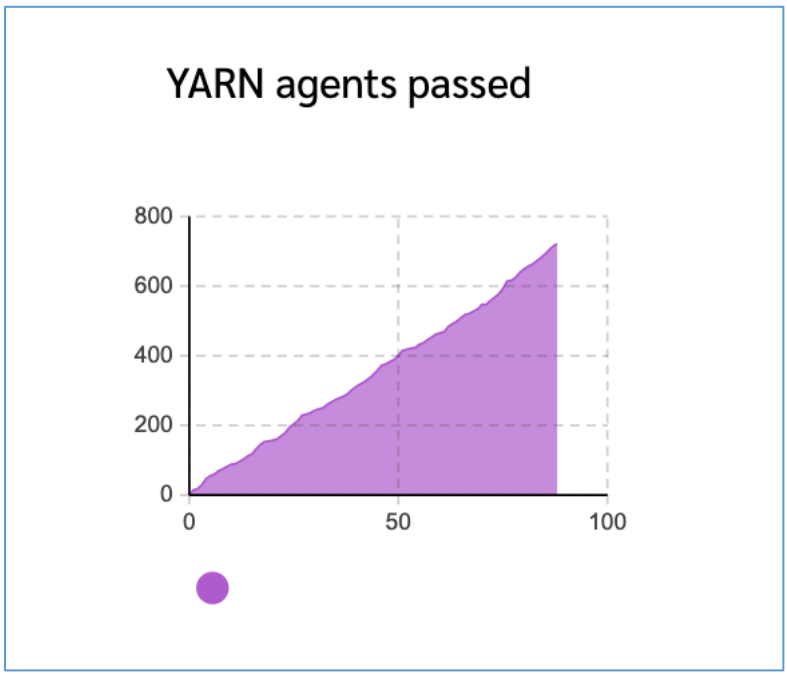


Рисунок 9 – Гистограммы

5. Добавьте на экран эксперимента и запуска модели элемент управления интенсивностью поступления агентов. Увеличьте в 2 раза интенсивность поступления агентов. Опишите словами, как изменятся результаты моделирования работы системы. Уменьшите в 2 раза по сравнению с исходным значением интенсивность поступления требований. Опишите словами, как изменятся результаты моделирования работы системы (1 балл).

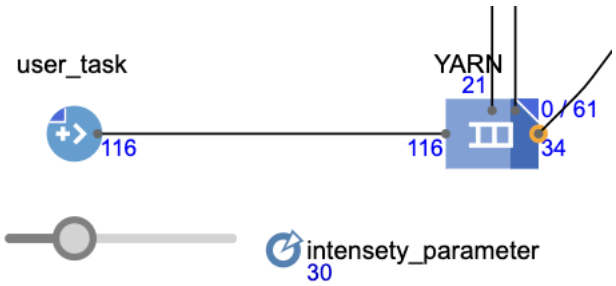
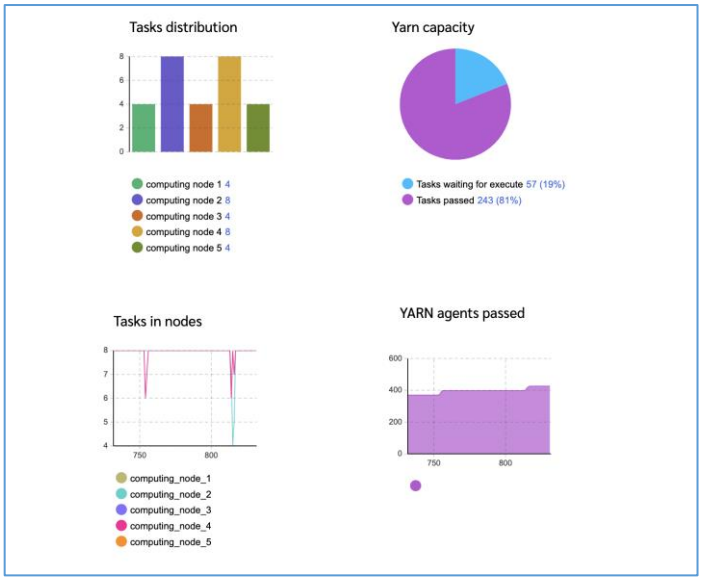
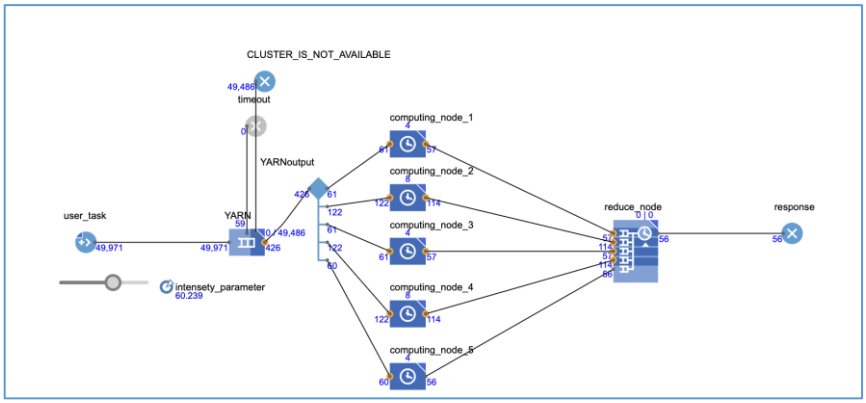


Рисунок 10 – Экран эксперимента и запуска модели с элементом управления интенсивностью поступления агентов





При увеличении интенсивности в 2 раза, кластер буквально перестал работать, из 50 тысяч задач, успешно отработали только 2, при этом как можно видеть из графиков, все computing_node заняты на 100%

Рисунок 11 – Работа модели при увеличении в 2 раза интенсивности поступления агентов

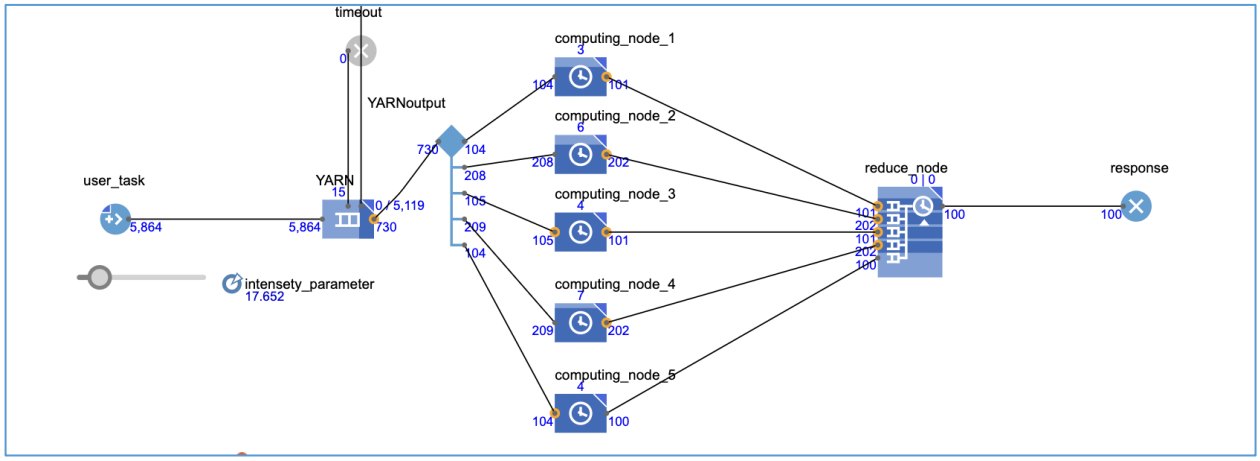


Рисунок 12 – Работа модели при уменьшении в 2 раза интенсивности поступления агентов

При уменьшении интенсивности в 2 раза, кластер работает также стабильно как и при дефолтной интенсивности – 30 задач в минуту, на скорость выполнения задач, работы системы не влияет, что говорит о том, что кластер имеет большую пропускную способность чем 15 задач в минуту, минимум 30.

6. Проведите эксперимент по варьированию минимум 2-х параметров модели и визуализируйте результаты. При визуализации отобразите значения варьируемых параметров и значения минимум 2-х результатов работы системы (2 балла).

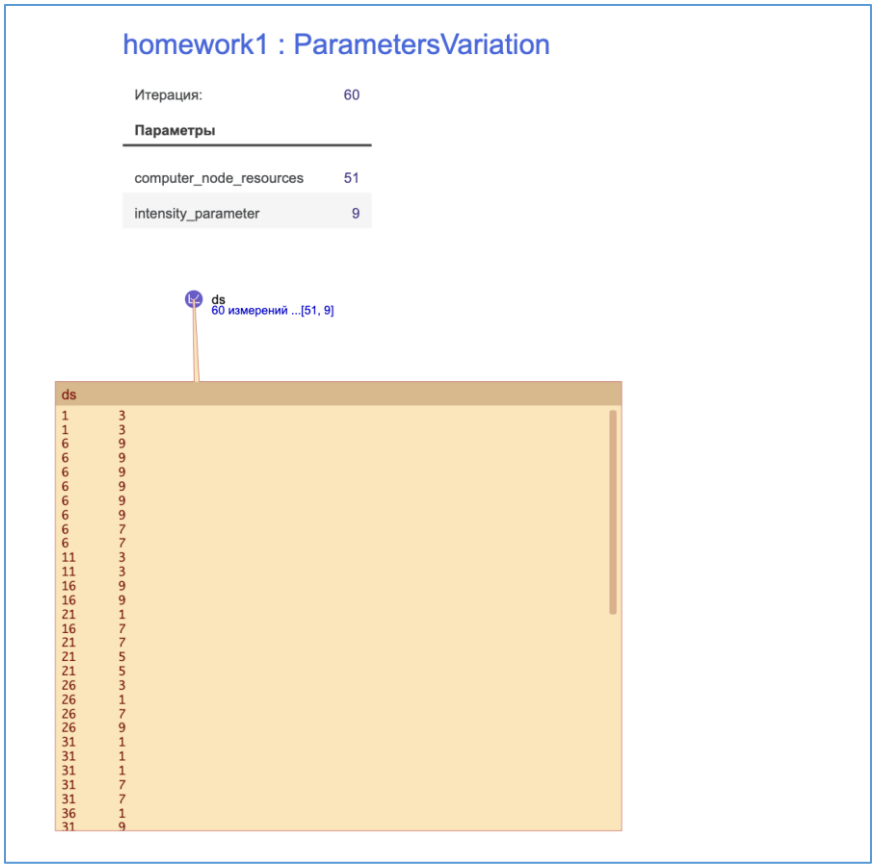


Рисунок 13 – Таблица AnyLogic параметров вариабельности параметров модели

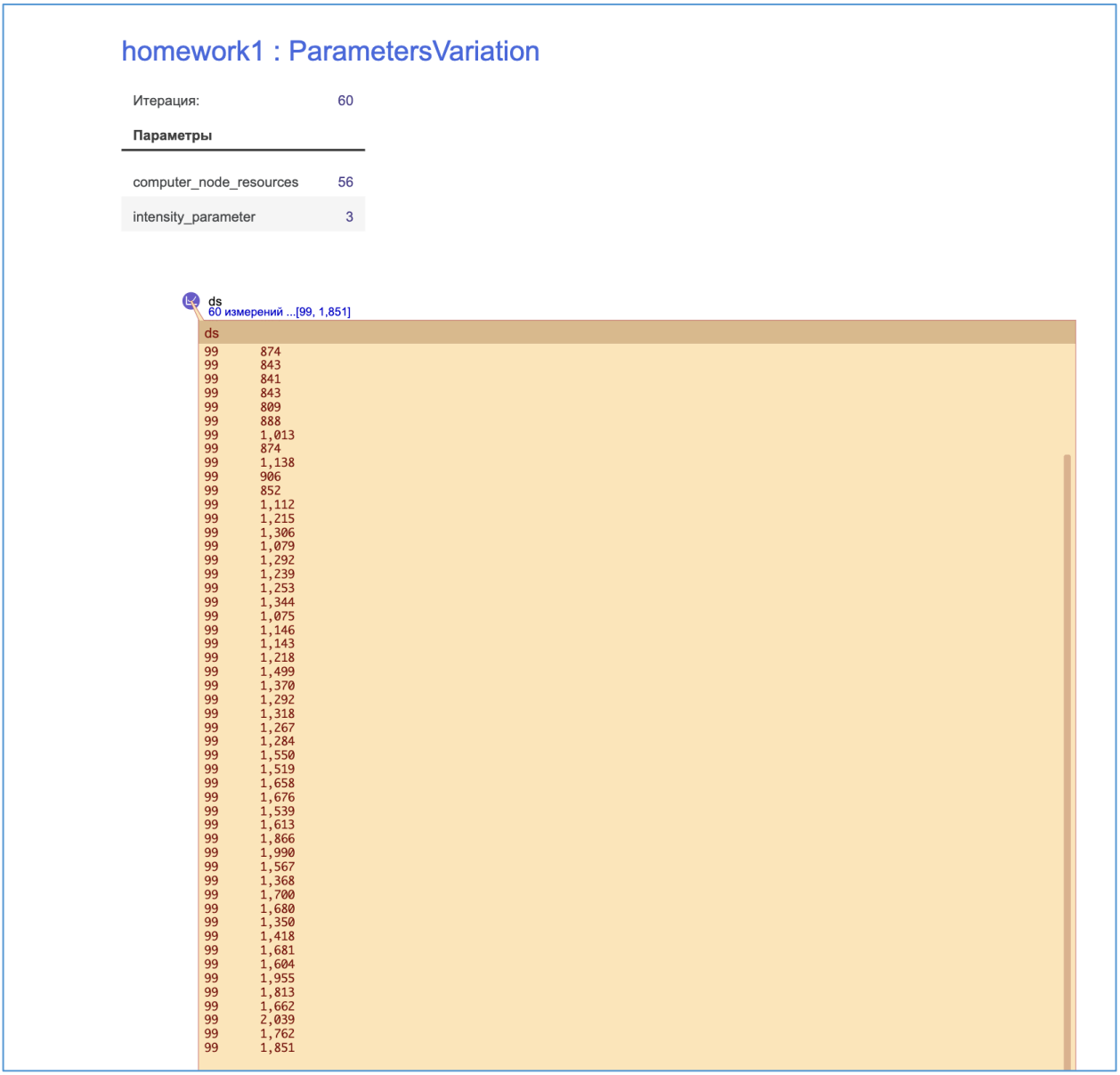
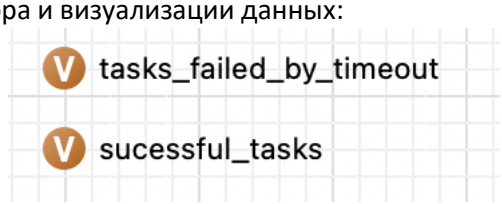


Рисунок 14 – Полученные/собранные данные в форме таблицы, отображающие значения варьируемых параметров и значений результатов работы системы

Действия Java для сбора и визуализации данных:



Действие после "прогона" модели:

```
ds.add(root.sucessful_tasks, root.tasks_failed_by_timeout);
```

7. Выберите целевую функцию, которая связана с потенциальной целью/назначением анализа системы. Проведите оптимизационный эксперимент. Опишите полученный результат (1 балл).

Целевая функция: root.sucessful_tasks/root.tasks_failed_by_timeout

Минимизация/максимизация целевой функции: максимизация

Параметры:

Параметр	Тип	Значение			
		Мин.	Макс.	Шаг	Начальное
intense...rameter	дискретный	15	60	5	
comput...ources	дискретный	1	10	2	

Рисунок 15 – Таблица AnyLogic параметров вариабельности параметров модели в эксперименте по оптимизации

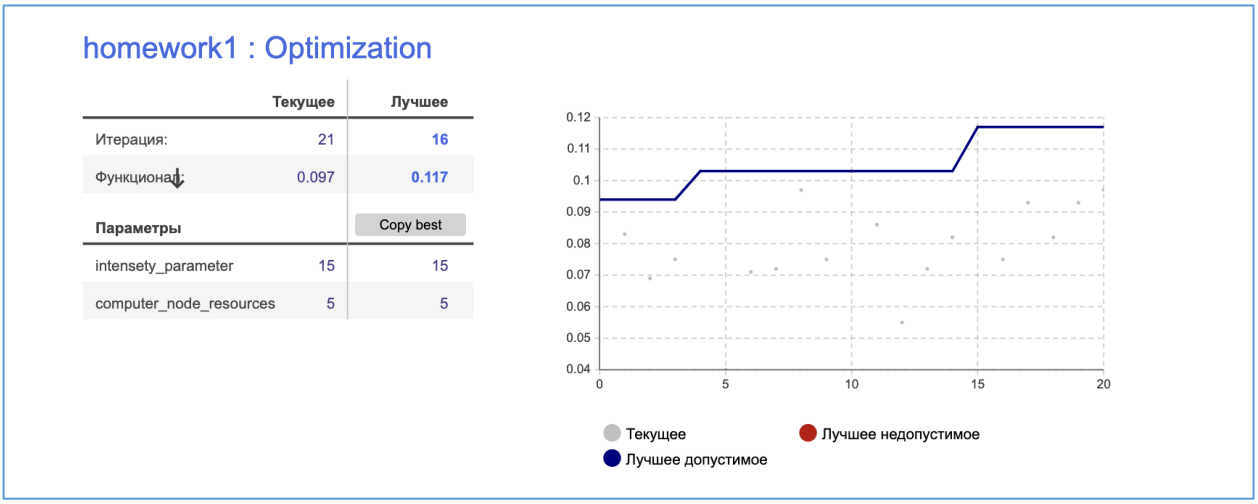


Рисунок 16 – Результаты эксперимента по оптимизации

Computer_node_resources – вычислительные мощности экзекуторов (влияют на скорость работы выполнения задач в computer_nodes)

Intensity_parameter – интенсивность задач (задач в минуту)

Выводы о результатах оптимизационного эксперимента: Эксперимент показал, что если увеличить вычислительные мощности ресурсов в 5 раз, то мы достигнем оптимального состояния работы кластера (при это нам необходимо увеличивать мощности ресурсов в 10 раз, достаточно только в 5), также видно что лучше всего при интенсивности в 15 задач в минуту, что очевидно, однако результаты эксперимента показали, что при интенсивности в 30 задач в минут, кластер продолжает стабильно работать => лучший шаг к оптимизации – увеличение мощностей до х5 раз.