



ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
"САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С. П. КОРОЛЕВА

(САМАРСКИЙ УНИВЕРСИТЕТ)

ИНСТИТУТ ИНФОРМАТИКИ, МАТЕМАТИКИ И ЭЛЕКТРОНИКИ

Факультет информатики

Кафедра программных систем

Дисциплина

Моделирование информационных процессов и систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе на тему

«МОДЕЛИРОВАНИЕ МНОГОТЕРМИНАЛЬНОЙ
ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ»

Вариант 9

Студент _____ Гижевская В.Д.

Руководитель _____ Баландин А.В.

Самара 2021

СОДЕРЖАНИЕ

Содержание	2
1. Описание исследуемой системы как предмета моделирования	3
1.1. Описание МВС	3
1.2. Стратегия сервера	4
2. Описание целей моделирования	6
3. Разработка математической модели	7
4. Вариант задания	12
5. Итоговая математическая модель	13
6. Разработка имитационной модели	14
6.1. Имитационная модель МВС	14
6.1.1. Терминальные станции	18
6.1.2. Сервер	23
6.1.3. Таймауты	26
6.2. Агент «Zadanie»	27
6.3. Сбор статистики и разделение потоков агентов	28
7. Оценка адекватности модели	30
8. Оптимизационный эксперимент	32
9. Эксперименты с моделью и статистическая оценка характеристик системы	35
10. Итоговые результаты и выводы	37
Список использованных источников	38

1. Описание исследуемой системы как предмета моделирования

1.1. Описание МВС

Предметом моделирования является многотерминальная вычислительная система (МВС) на базе локальной вычислительной сети (Рисунок 1), которая состоит из центрального компьютера (сервера) и взаимодействующих с ним в интерактивном режиме терминальных станций (ТС), используемых операторами для подготовки вычислительных заданий, отправки их на выполнение серверу и получения результатов. МВС работает под управлением сетевой многопроцессной операционной системы, обеспечивающей взаимодействие между процессами-клиентами ТС, распределёнными по узлам локальной сети, и процессом-сервером посредством обмена сообщениями.

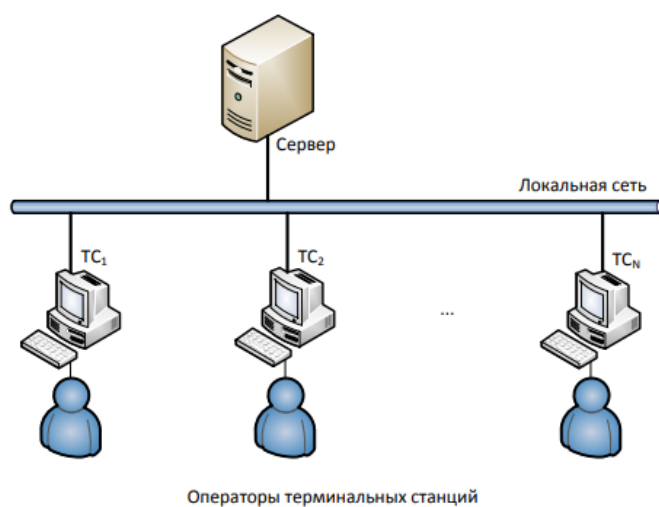


Рисунок 1 – Многотерминальная вычислительная сеть

Многотерминальная система состоит из сервера и N терминальных станций (ТС), подключённых к локальной сети. Терминальные станции используются операторами для подготовки вычислительных заданий, отправляемых серверу для выполнения. Подготовленное оператором ТС

вычислительное задание отправляется серверу для выполнения, после чего оператор ожидает получение результатов. Результаты выполненного вычислительного задания сервер отправляет на ТС оператора. После получения результатов и их анализа оператор подготавливает и отправляет серверу очередное задание. Далее всё повторяется. Качество работы МВС оценивается оператором по длительности времени ожидания получения результатов выполнения задания.

Вычислительное задание формируется оператором на терминальной станции посредством текстового редактора в виде командных строк, предназначенных для их последовательного выполнения командным интерпретатором сервера. Оператор, завершив подготовку задания, автоматически инициирует на ТС программный процесс-клиент, реализующий транзакцию отправки задания по локальной сети серверу в виде текстового сообщения и приёма от сервера результатов выполнения. При этом процесс, реализующий транзакцию, выдаёт на экран оператора ТС сообщения о ходе её выполнения. Если сервер занят выполнением некоторого ранее принятого задания, то ТС ожидает начала приёма задания сервером.

1.2. Стратегия сервера

Сервер может реализовывать различные варианты стратегий учёта инициируемых ТС транзакций и мест нахождения заданий, ожидающих выполнения сервером. Стратегия сервера данного в задании варианта представлены ниже в таблице:

Вариант №	Стратегия сервера по обслуживанию транзакции
<i>Вариант 4</i>	Сервер имеет накопитель на $N < K$ транзакций. Инициированная ТС транзакция принимается на обслуживание, если накопитель не полон. Задание принятой на обслуживание транзакции либо сразу поступает на

	<p>выполнение, либо принимается в накопитель и ставится в очередь заданий в порядке поступления.</p> <p>Если накопитель полон, то транзакция получает отказ в обслуживании и аннулируется. ТС, получившая отказ, выдерживает паузу (таймаут) и вновь инициирует транзакцию, продолжая действовать так до тех пор, пока транзакция не будет принята на обслуживание, после чего ТС ждёт сообщения с результатом выполненного задания, а после его получения выдаёт результат на экран оператору, и транзакция завершается.</p>
--	---

Время выполнения сервером любого принятого на обработку вычислительного задания, складывается из времён последовательного выполнения сервером команд задания. Время выполнения сервером команды любого задания – τ_{cmd} , является случайным в диапазоне $[\tau_{cmd}^{min}, \tau_{cmd}^{max}]$. Завершив выполнение очередного задания, сервер отправляет процессу-клиенту соответствующей ТС сообщения с результатами, которые выдаются на экран оператору. Получив результаты выполненного задания и осуществив их анализ, оператор приступает к подготовке следующего задания, после чего цикл подготовки оператором очередного задания и запуск транзакции по его выполнению повторяется. Время анализа оператором полученного результата выполнения задания укладывается в диапазон $[\tau_{res}^{min}, \tau_{res}^{max}]$.

2. Описание целей моделирования

Целью моделирования МВС является определение характеристик оперативной обработки сервером вычислительных заданий, поступающих с терминальных станций, и нахождение рациональных параметров системы, обеспечивающих операторам работу в режиме online (время ожидания ответа укладывается в диапазон $1 \div 3$ минуты).

Для заданной стратегии обработки очереди принятых заданий сервером необходимо оценить время ожидания оператором результатов выполнения задания, а также соотношение в нём времён ожидания начала обработки задания и времени выполнения задания сервером. Для этого необходимо построить модель МВС как системы обслуживания вычислительных заданий и с её помощью в зависимости от заданных параметров формирования потока транзакций от терминальных станций и параметров обслуживания вычислительных заданий сервером МВС оценить следующие характеристики её работы:

- Соотношение времён занятости сервером выполнением вычислительных заданий и времени ожидания сервером транзакций от ТС;
- Время ожидания i -ой ТС начала обслуживания транзакции на выполнение вычислительного задания сервером;
- Время ожидания оператором i -ой ТС результатов выполнения задания;
- Пропускная способность сервера;
- Производительность оператора i -ой ТС;
- Эффективность загрузки сервера;
- Эффективность работы оператора i -ой ТС.

3. Разработка математической модели

Для сервера МВС, как прибора обслуживания, важным параметром является интенсивность инициации терминальными станциями транзакций на выполнение вычислительных работ. Интенсивность потока транзакций, поступающих серверу от i -ой ТС в замкнутой МВС зависит от величины случайно формируемого интервала времени иницирования транзакций оператором ТС в секундах – $\tau_i^{оп}$, а также от величины таймаута в секундах – τ_{out}^{tr} автоматической повторной инициации транзакции ТС при отказе занятого обработкой сервера принять её на обслуживание.

Интенсивность потока транзакций от i -ой ТС зависит от параметров формирования интервала времени иницирования транзакций оператором i -ой ТС и случайного количества r таймаутов повторной инициации терминальной станцией транзакции, получившей отказ в обслуживании – $\tau_i^{ТС} = \tau_i^{ТС}(\tau_i^{оп}, r \cdot \tau_{out}^{tr})$, которое является эндогенной случайной величиной, зависящей от времени формирования оператором задания – τ_{task} , времени ожидания результата – τ_{wait} , и времени анализа оператором результата – τ_{res} . Все эти параметры являются случайными.

Значение $\tau_{task}(k_{cmd}, \tau_{cmd})$ – эндогенный случайный параметр, измеряемый в секундах, который зависит от количества k_{cmd} сформированных в задании команд, $k_{cmd} = 1, 2, 3, \dots$ – случайное целое число, и от τ_{cmd} – случайный непрерывный интервал времени формирования оператором одной командной строки, измеряемый в секундах, $\tau_{cmd} \in [\tau_{cmd}^{min}, \tau_{cmd}^{max}]$. Очевидно, $\tau_{task}(k_{cmd}, \tau_{cmd}) = k_{cmd} \cdot \tau_{cmd}$. Тогда среднее время формирования задания оператором абстрактной ТС в секундах будет равно $M[\tau_{task}] = M[k \cdot \tau_{cmd}]$.

Время ожидания результата оператором абстрактной ТС – τ_{wait} , является непрерывной случайной величиной, измеряемой в секундах, которая при заданной дисциплине обслуживания транзакций сервером зависит от его производительности.

Время анализа оператором абстрактной ТС результатов выполнения задания – τ_{res} , является непрерывной случайной величиной.

С учётом введенных обозначений выражение для величины интервала (в секундах) между инициациями транзакций в потоке от ТС – τ^{TC} , примет вид: $\tau^{TC} = k \cdot \tau_{cmd} + \tau_{wait} + \tau_{res}$.

Интенсивность потока транзакций от абстрактной ТС к серверу МВС зависит от среднего значения интервала времени поступления заявок в секундах:

$$M[\tau^{TC}] = M[k_{cmd} \cdot \tau_{cmd} + \tau_{wait} + \tau_{res}].$$

Допустимо положить, что случайные величины k_{cmd} , τ_{cmd} , τ_{res} являются независимыми, и в установившемся режиме работы МВС их математические ожидания не зависят от времени. Тогда математическое ожидание эндогенного случайного параметра τ_{wait} в установившемся режиме работы МВС также не будет зависеть от времени, а следовательно:

$$M[\tau^{TC}] = M[k_{cmd}] \cdot M[\tau_{cmd}] + M[\tau_{wait}] + M[\tau_{res}].$$

Будем полагать, среднее время (в секундах) выполнения сервером любой команды любого задания любой ТС одно и то же – $M[\tau_{cmd}^i] = M[\tau_{cmd}^j] = M[\tau_{cmd}]$, а также среднее время (в секундах) анализа результата одинаково для операторов любой станции – $M[\tau_{res}^i] = M[\tau_{res}^j] = M[\tau_{res}]$. А вот случайное количество команд, формируемое в задании операторами разных ТС, и среднее время ожидания ответа от сервера в установившемся режиме работы МВС различны для операторов разных ТС. Обозначим интенсивность потока транзакций от i -ой ТС как λ_i , тогда:

$$\lambda_i = \frac{1}{M[\tau_i^{TC}]} = \frac{1}{M[k_{cmd}^i \cdot \tau_{cmd} + \tau_{wait}^i + \tau_{res}]}.$$

Если положить, что в установившемся режиме работы МВС средние значения всех случайных параметров не зависят от времени то:

$$\lambda_i = \frac{1}{M[\tau_i^{TC}]} = \frac{1}{M[k_{cmd}^i] \cdot M[\tau_{cmd}] + M[\tau_{wait}^i] + M[\tau_{res}]},$$

а интенсивность потока заявок, поступающих серверу со всех ТС будет:

$$\Lambda = \sum_{i=1}^N \lambda_i ,$$

где N – количество терминальных станций. Если положить, что все ТС формируют транзакции с одинаковой интенсивностью λ , то выражение примет вид:

$$\Lambda = N\lambda.$$

Интенсивность потока транзакциями, поступающих серверу от каждой ТС:

$$\Lambda = \frac{N}{M[\tau_{TC}]} .$$

Внутренним параметром сервера является его вычислительная производительность, характеризующаяся средним количеством команд вычислительного задания, выполняемых сервером в единицу времени – μ . Так как τ_{srv} не зависит от того, какому заданию и какой ТС команды принадлежат, то:

$$\mu = \frac{1}{M[\tau_{srv}]} .$$

В качестве выходных характеристик, оценивающих эффективность работы МВС в соответствии с поставленными целями, введём в рассмотрение следующие эндогенные параметры:

- τ_{wait}^i – время ожидания результата выполнения задания оператором i -ой ТС в секундах;
- τ_{trans}^i – время ожидания i -ой ТС начала обслуживания транзакции сервером в секундах;
- P_{TC}^i – производительность i -ой ТС, измеряемая в $\frac{\text{шт}}{\text{с}}$;
- abs – абсолютная пропускная способность, измеряемая в $\frac{\text{шт}}{\text{с}}$;
- ren^i – вероятность отказа в обслуживании задания, инициализированного i -ой ТС;
- $W_{сервер}$ – отношение времени занятости сервера выполнением

заданий ко времени ожидания сервером транзакций;

- W_{TC}^i – отношение времени занятости i -ой ТС ко времени её простоя;
- $Q_{сервер}$ – эффективность работы сервера, являющаяся безразмерной величиной;
- Q_{TC}^i – эффективность работы оператора i -ой ТС, являющаяся безразмерной величиной.

Время ожидания i -ой ТС начала обслуживания транзакции сервером будет складываться из времени (в секундах), которое задание находится в очереди τ_{queue}^i , зависящего от количества заданий, находящихся в очереди, и величины суммарного таймаута $r * \tau_{out}^{tr}$: $\tau_{trans}^i = \tau_{queue}^i + r * \tau_{out}^{tr}$, где r – случайное количество таймаутов.

Производительность i -ой ТС вычисляется как среднее количество инициализированных i -ой ТС транзакций n_{init}^i в единицу времени: $P_{TC}^i = \frac{n_{init}^i}{t}$, где t – время.

Абсолютную пропускную способность можно найти как среднее количество завершённых системой заданий $n_{finished} = \sum n_{finished}^i$ в единицу времени: $abs = \frac{n_{finished}}{t} = \frac{\sum n_{finished}^i}{t}$, где $n_{finished}^i$ – количество завершённых заданий, инициализированных i -ой ТС.

Относительная пропускная способность будет равна отношению количества завершённых системой заданий к суммарному количеству инициализированных: $rel = \frac{\sum n_{finished}^i}{\sum n_{init}^i}$.

Вероятность отказа в обслуживании транзакций i -ой ТС находится как отношение количества инициализированных i -ой ТС транзакций, которым было отказано в обслуживании n_{ren}^i к общему количеству инициализированных i -ой ТС транзакций: $ren^i = \frac{n_{ren}^i}{n_{init}^i}$.

Отношение времени занятости сервера выполнением заданий ко

времени ожидания сервером транзакций по определению может быть

найдено как:
$$W_{\text{сервер}} = \frac{\sum(\tau_{\text{wait}}^i - \tau_{\text{queue}}^i)}{t - \sum(\tau_{\text{wait}}^i - \tau_{\text{queue}}^i)}.$$

Отношение времени занятости i-ой ТС ко времени её простоя, по

анalogии:
$$W_{\text{ТС}}^i = \frac{k_{\text{cmd}}^i \cdot \tau_{\text{cmd}} + \tau_{\text{res}}}{t - (k_{\text{cmd}}^i \cdot \tau_{\text{cmd}} + \tau_{\text{res}})}.$$

Эффективность работы i-ой ТС будем оценивать как отношение времени, которое задание находится на ТС (т.е. сумма времени формирования транзакции и времени анализа завершённой транзакции), к

общему времени работы системы:
$$Q_{\text{ТС}}^i = \frac{k_{\text{cmd}}^i \cdot \tau_{\text{cmd}} + \tau_{\text{res}}}{t}.$$

Аналогично, эффективность работы сервера оценим как отношение времени, которое сервер занят выполнением заданий, к общему времени

работы системы:
$$Q_{\text{сервер}} = \frac{\sum(\tau_{\text{wait}}^i - \tau_{\text{queue}}^i)}{t}.$$

4. Вариант задания

В рамках варианта задания №9 были использованы следующие параметры:

- Количество ТС $N = 5$;
- Среднее количество команд в задании $M[k_{cmd}] = 10$ шт;
- Распределение $F(k_{cmd})$ – экспоненциальное $exponential(\frac{1}{10}, 1)$;
- Среднее время формирования командной строки $M[\tau_{cmd}] = 48$ сек;
- Распределение $F(k_{cmd})$ – треугольное $triangular(36, 60)$;
- Таймаут $\tau_{out}^{tr} = 48$ сек;
- Среднее время анализа результата $M[\tau_{res}] = 210$ сек;
- Распределение $F(\tau_{res})$ – равномерное $uniform(150, 270)$;
- Среднее время выполнения команды $M[\tau_{srv}] = 4.8$ сек;
- Распределение $F(\tau_{srv})$ – нормальное $normal(0.3, 4.8)$;

При этом параметры распределений, не указанные в варианте задания явно, подбирались вручную.

5. Итоговая математическая модель

По итогам моделирования имеем следующую математическую модель:

$$M_{MBC} = R_{MBC} \left\{ \begin{array}{l} \tau_{trans}^i = \tau_{queue}^i + r * \tau_{out}^{tr} \\ P_{TC}^i = \frac{n_{init}^i}{t} \\ abs = \frac{n_{finished}}{t} \\ n_{finished} = \sum n_{finished}^i \\ rel = \frac{\sum n_{finished}^i}{\sum n_{init}^i} \\ Q_{TC}^i = \frac{k_{cmd}^i * \tau_{cmd} + \tau_{res}}{t} \\ Q_{сервер} = \frac{\sum (\tau_{wait}^i - \tau_{queue}^i)}{t} \\ ren^i = \frac{n_{ren}^i}{n_{init}^i} \\ W_{сервер} = \frac{\sum (\tau_{wait}^i - \tau_{queue}^i)}{t - \sum (\tau_{wait}^i - \tau_{queue}^i)} \\ W_{TC}^i = \frac{k_{cmd}^i * \tau_{cmd} + \tau_{res}}{t - (k_{cmd}^i * \tau_{cmd} + \tau_{res})} \\ \tau_{task} = k_{cmd} * \tau_{cmd} \\ M[\tau_{task}] = M[k * \tau_{cmd}] \\ \tau^{TC} = k * \tau_{cmd} + \tau_{wait} + \tau_{res} \\ M[\tau^{TC}] = M[k_{cmd}] * M[\tau_{cmd}] + M[\tau_{wait}] + M[\tau_{res}] \\ \lambda_i = \frac{1}{M[\tau_i^{TC}]} \\ \Lambda = \sum_{i=1}^N \lambda_i = N\lambda \\ \mu = \frac{1}{M[\tau_{srv}]} \\ N = 5 \\ M[k_{cmd}] = 10 \\ F(k_{cmd}) = \text{exponential}(\frac{1}{10}, 1) \\ M[\tau_{cmd}] = 48 \\ F(\tau_{cmd}) = \text{triangular}(36, 60) \\ \tau_{out}^{tr} = 48 \\ M[\tau_{res}] = 210 \\ F(\tau_{res}) = \text{uniform}(150, 270) \\ M[\tau_{srv}] = 4.8 \\ F(\tau_{srv}) = \text{normal}(0.3, 4.8) \\ t = \text{time}() \end{array} \right.$$

6. Разработка имитационной модели

6.1. Имитационная модель MVC

Результаты разработки имитационной модели MVC, соответствующей варианту задания, представлены на рисунках 2-6.

На рисунке 2 представлена имитационная модель самой моделируемой MVC, состоящая из трёх основных частей, описываемых в разделах 6.1.1 – 6.1.3:

- Терминальные станции (ТС)
- Сервер
- Таймауты

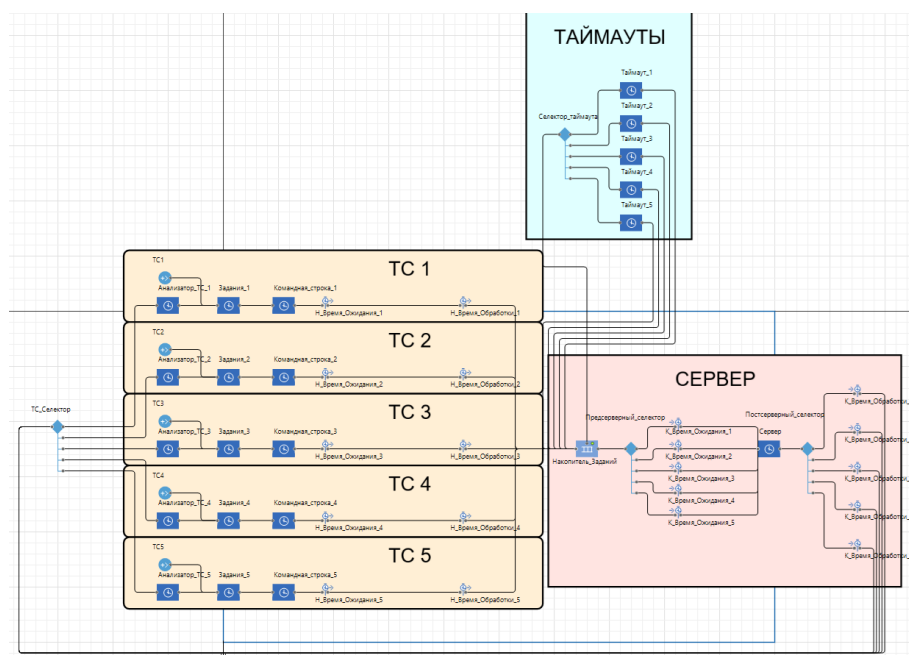


Рисунок 2 – Имитационная модель MVC

На рисунке 3 представлены переменные, созданные для сбора статистики, а также параметры, необходимые для проведения оптимизационного эксперимента:

- Заданий_TC_1 – Заданий_TC_5 – содержат количество

выполненных заданий, инициализированных каждой ТС;

- Отказы_ТС_1 – Отказы_ТС_5 – содержат количество отказов в обслуживании для каждой ТС;
- Время_активной_работы_ТС_1 – Время_активной_работы_ТС_5 – содержат суммарное время активной работы каждой ТС, т.е время, которое агенты находятся на каждой из ТС;
- Эффективность_ТС_1 – Эффективность_ТС_5 – содержат эффективность каждой ТС;
- Эффективность_сервера – содержит эффективность сервера;
- Продуктивность_ТС_1 – Продуктивность_ТС_5 – содержат продуктивность каждой ТС;
- Всего_обработано – содержит общее количество обработанных МВС заданий;
- Всего_создано – содержит общее количество инициализированных всеми ТС заданий;
- Абсолютная_ПС – абсолютная пропускная способность системы;
- Относительная_ПС – относительная пропускная способность системы;
- Время_активной_работы_сервера – содержит общее время, которое сервер обрабатывает задания;
- С_Время_работы_ко_времени_простоя – содержит отношение времени, которое сервер обрабатывает задания, ко времени, которое сервер ожидание поступления заданий;
- Вместимость_накопителя – длина очереди;
- ТС_1_Время_работы_ко_времени_простоя –
ТС_5_Время_работы_ко_времени_простоя – отношения времени занятости каждой ТС ко времени ее простоя.
- Вероятность_отказа_ТС_1 – Вероятность_отказа_ТС_5 –
вероятности отказа в обслуживании задания,

инициализированного каждой ТС.

Вероятность отказа в обслуживании будем высчитывать как отношение количества инициализированных i -ой ТС заданий, которым было отказано в рамках содержащих их транзакций, к общему числу инициализированных i -ой ТС заданий

✓ Заданий_TC_1	✓ Отказы_TC_1	✓ Время_активной_работы_TC_1
✓ Заданий_TC_2	✓ Отказы_TC_2	✓ Время_активной_работы_TC_2
✓ Заданий_TC_3	✓ Отказы_TC_3	✓ Время_активной_работы_TC_3
✓ Заданий_TC_4	✓ Отказы_TC_4	✓ Время_активной_работы_TC_4
✓ Заданий_TC_5	✓ Отказы_TC_5	✓ Время_активной_работы_TC_5
		✓ Время_активной_работы_сервера
✓ Эффективность_TC_1	✓ Продуктивность_TC_1	✓ Всего_обработано
✓ Эффективность_TC_2	✓ Продуктивность_TC_2	✓ Всего_создано
✓ Эффективность_TC_3	✓ Продуктивность_TC_3	
✓ Эффективность_TC_4	✓ Продуктивность_TC_4	✓ Относительная_ПС
✓ Эффективность_TC_5	✓ Продуктивность_TC_5	⚙ Абсолютная_ПС
⚙ Эффективность_Сервера	⚙ Вместимость_Накопителя	
	⚙ С_Время_работы_ко_времени_простоя	
✓ Вероятность_отказа_TC_1	⚙ TC_1_Время_работы_ко_времени_простоя	
✓ Вероятность_отказа_TC_2	⚙ TC_2_Время_работы_ко_времени_простоя	
✓ Вероятность_отказа_TC_3	⚙ TC_3_Время_работы_ко_времени_простоя	
✓ Вероятность_отказа_TC_4	⚙ TC_4_Время_работы_ко_времени_простоя	
✓ Вероятность_отказа_TC_5	⚙ TC_5_Время_работы_ко_времени_простоя	

Рисунок 3 – Переменные и параметры для сбора статистики

Гистограммы, содержащие статистику времени ожидания каждой из ТС обслуживания транзакции и времени ожидания операторами ТС результатов выполнения транзакций представлены на рисунках 4 и 5

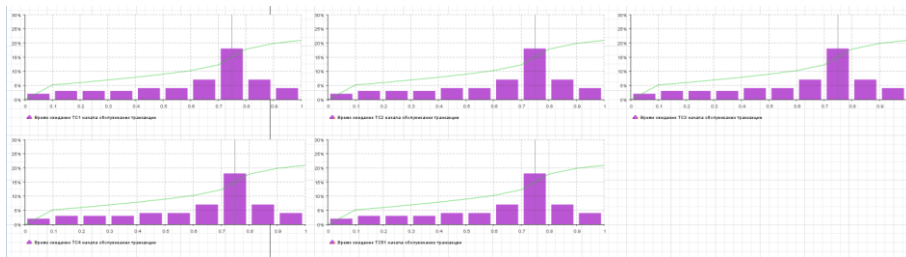


Рисунок 4 – Гистограммы времени ожидания ТС начала обслуживания транзакций

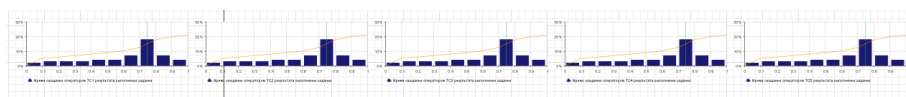


Рисунок 5 – Гистограммы времени ожидания операторами ТС результатов выполнения задания

На рисунке 6 представлены следующие столбчатые диаграммы:

- диаграмма эффективности сервера;
- диаграммы абсолютной и относительной пропускных способностей системы;
- диаграмма отношения времени занятости сервера выполнением заданий ко времени ожидания сервером транзакций;
- диаграммы отношений времени занятости каждой ТС ко времени ее простоя;
- диаграммы эффективности каждой ТС;
- диаграммы производительности каждой ТС;
- диаграммы вероятностей отказа для каждой ТС.



Рисунок 6 – Столбчатые диаграммы

6.1.1. Терминальные станции

Первая часть имитационной модели, имитирующая работу терминальных станций, состоит из пяти блоков Source под названиями TC1 – TC5, задачей которых является создание пяти агентов «Zadanie» в момент начала имитационного эксперимента и присвоение их внутреннему параметру Номер_ТС значения, равному номеру ТС, в рамках которой они создаются. С этой целью глобальный агент Main в момент запуска имитационного эксперимента вызывает для каждого Source функцию inject(), тем самым обеспечивая однократную инициализацию агентов «Zadanie» внутри этих объектов.

На рисунках 8 и 9 представлены, соответственно, свойства одного из блоков Source и свойства агента Main.

+

TC1 - Source

Имя:

TC1

☒ Отображать имя

☐ Исключить

Прибывают согласно:

Вызовом функции inject()

Местоположение прибытия:

Не задано

Агент

Специфические

Описание

Рисунок 8 – Свойства блока ТС

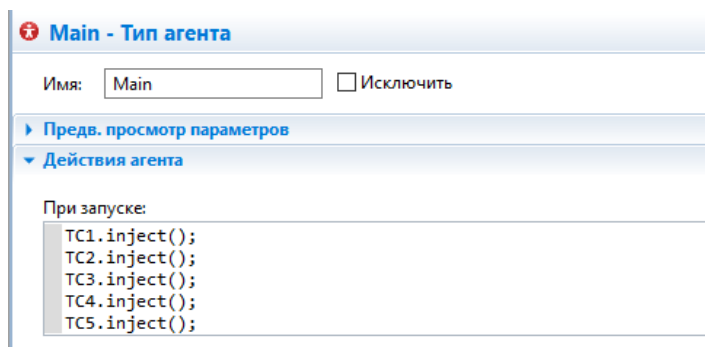


Рисунок 9 – Свойства агента Main

Далее расположены пять пар по два блока Delay. Первый блок Delay (Задания_1 – Задания_5) служит для случайного определения количества команд в задании и не имеет времени задержки. Также он устанавливает значение внутреннего параметра агента «Отказано» на false, что необходимо для отслеживания отказов в обработке. В дополнение, этот объект изменяет значения следующих переменных модели и внутренних параметров агента, необходимых для сбора статистики о работе системы:

- Время_в_системе;
- Таймер;
- Всего_создано;
- Заданий_TC_X (где X – номер терминальной станции).

Второй блок Delay (Командная_строка_1 – Командная_строка_5) имитирует ввод оператором ТС команд и поэтому имеет задержку, формируемую функцией Ф_Командная_строка путем суммирования в цикле значений случайных величин, подчиняющихся треугольному распределению. Эта функция принимает на вход количество строк кода в задании и возвращает сумму случайных величин из заданного распределения.

На рисунке 10 представлена реализация функции Ф_Командная_строка.

Ф_Командная_строка - Функция

Имя: ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☐ Действие (не возвращает ничего)

☒ Возвращает значение

Тип:

Аргументы

Имя	Тип
count	int

Тело функции

```
double result = 0;
for (int i = 0; i < count; i++){
    result+=triangular(36,60);
}
return result;
```

Рисунок 10 – Функция Ф_Командная_строка

Также объект Командная_строка фиксирует во внутренней переменной агента Время_внутри_ТС время пребывания внутри данного блока с целью сбора данных о времени, которое агент проводит на терминальной станции.

На рисунках 11 и 12 представлены свойства блоков Задания и Командная_строка.

Добавлено примечание ([A.B1]): Почему задержка на 0, а не очередь на 1? Или одна задержка с действиями на входе, на выходе.

Добавлено примечание ([ГВД2R1]): Этот блок необходим по большому счёту для генерации числа строчек кода в задании, поэтому на нём нет задержки. А во втором блоке нельзя это сделать, потому что блок Delay работает следующим образом. Вычисляется время задержки -> выполняются действия "При входе" -> Сама задержка. Следовательно, мы не можем рассчитать там количество строк, потому что вычисление времени задержки идёт перед тем, что выполнится при входе

⌚

Задания_1 - Delay

Имя:

Задания_1

☒ Отображать имя
 ☐ Исключить

Тип задержки:

☒ Определенное время
☐ До вызова функции stopDelay()

Время задержки:

0

секунды

Вместимость:

1

Максимальная вместимость:

Место агентов:

Специфические

Действия

При входе:

```
agent.Строки_кода=(int)(exponential(0.1,1));
agent.Отказано = false;
```

При подходе к выходу:

При выходе:

```
agent.Время_в_системе=time(MINUTE);
agent.Таймер=time(MINUTE);
Всего_создано++;
Заданий_TC_1++;
```

При извлечении:

Специфические

Описание

Рисунок 11 – Свойства блока Задания

⌚

Командная_строка_1 - Delay

Имя:

Командная_строка_1

☒ Отображать имя

☐ Исключить

Тип задержки:

☒ Определенное время

☐ До вызова функции stopDelay()

Время задержки:

секунды

Вместимость:

Максимальная вместимость:

Место агентов:

📍

▼ Специфические

Выталкивать агентов:

☒

Вернуть агента в исходную точку:

☒

Включить сбор статистики:

☐

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

Добавлено примечание ([А.ВЗ]): Выражение задержки не описано, параметры распределения не мотивированы, время то в секунда, то в минутах, это почему?

Добавлено примечание ([ГВД4R3]): Это из-за того, что при производительности в секунду, получаются очень маленькие значения, с которыми не так удобно работать и анализировать, поэтому пропускные способности и производительность в минуту считаются

Рисунок 12 – Свойства блока Командная строка

Наконец, в данной части модели присутствуют пять блоков Delay под названиями Анализатор_TC_1 – Анализатор_TC_5, имитирующие анализ оператором ТС результатов выполнения задания. Аналогично объектам Delay из других блоков, в этих объектах собирается часть необходимых статистических данных, таких как:

- время прохождения агентов цикла в системе, записываемое в параметр агента `Время_в_системе`;
- время активной работы ТС, записываемое в параметр агента `Время_активной_работы_TC_X`, где `X` – номер соответствующей ТС;
- эффективность и продуктивность ТС, записываемые в переменные `Эффективность_TC_X` и `Продуктивность_TC_X`, где `X` – номер соответствующий ТС.

На рисунке 13 представлены свойства одного из блоков Анализатор.

от ТС в виде транзакций.

Вместимость накопителя может быть изменена при помощи параметра Вместимость_Накопителя.

Объект Сервер, помимо имитации задержки на обработку заданий, также фиксирует во внутреннем параметре агента Время_на_сервере время обработки задания, необходимое для последующего вычисления эффективности сервера.

Время задержки на сервере определяется при помощи функции Ф_Сервер, принимающей на вход количество строк кода в задании и возвращающей сумму значений случайных величин из нормального распределения.

Реализация функции Ф_Сервер представлена на рисунке 14.

Ф_Сервер - Функция

Имя:

☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☐ Действие (не возвращает ничего)

☒ Возвращает значение

Тип:

Аргументы

Имя	Тип
count	int

Тело функции

```
double result = 0;
for (int i = 0; i < count; i++){
    normal(0.3,4.8);
}
return result;
```

Рисунок 14 – Функция Ф_Сервер


Свойства этих блоков представлены на рисунках 15 и 16.

Накопитель_Заданий - Queue

Имя: ☒ Отображать имя ☐ Исключить

Вместимость:

Максимальная вместимость:

Место агентов: 

▼ Специфические

Очереди:

Разрешить уход по таймауту: ☐

Разрешить вытеснение: ☒

Вернуть агента в исходную точку: ☐

Включить сбор статистики: ☐

Рисунок 15 – Свойства блока Накопитель_Заданий

Сервер - Delay


Имя: ☒ Отображать имя ☐ Исключить

Тип задержки: ☒ Определенное время ☐ До вызова функции stopDelay()

Время задержки:

Вместимость:

Максимальная вместимость:

Место агентов: 

▼ Специфические

Выпалкивать агентов: ☐

Вернуть агента в исходную точку: ☒

Включить сбор статистики: ☐

▼ Действия

При входе:

При подходе к выходу:

При выходе:

При извлечении:

Рисунок 16 – Свойства блока Сервер

6.1.3. Таймауты

Часть модели, имитирующая таймауты, возникающие при отказе транзакции в обслуживании сервером, представляет собой пять блоков Delay под названиями Таймаут_1 – Таймаут_5, которые имитируют задержку, возникающую при нехватке места в накопителе и возникающих при этом отказах в обслуживании транзакций.

Т.к. при возникновении отказа в обслуживании транзакция аннулируется, а содержащееся в ней задание попадает в новую транзакцию, нам необходимо будет учесть этот момент следующим образом: в агенте "Zadanie" будем хранить параметр типа bool "Отказано", в который будем записывать true при первом прохождении агентом с текущим заданием блока таймаута. Дальнейшие прохождения того же самого задания через блок таймаута при помощи проверки на значение параметра "Отказано" учитывать не будем, т.к. задание не изменилось, оно лишь попало в другую транзакцию. Без этой проверки может возникнуть ситуация, при которой последовательно получат отказ транзакции на выполнение одного и того же задания, все разы будут зафиксированы и вероятность отказа, исходя из формулы подсчета вероятности отказа, описанной в разделе с математической моделью, превысит единицу, т.к. количество отказов превысит количество инициализаций заданий.

Также эти объекты участвуют в сборе статистики по отказам путём сбора их количества и подсчёта вероятностей отказа для каждой ТС.

Свойства одного из объектов Таймаут представлены на рисунке 17.

Добавлено примечание ([A.B6]): Отказ в обслуживании последовательно получают разные инициации транзакций на выполнение одного и тем же задания.

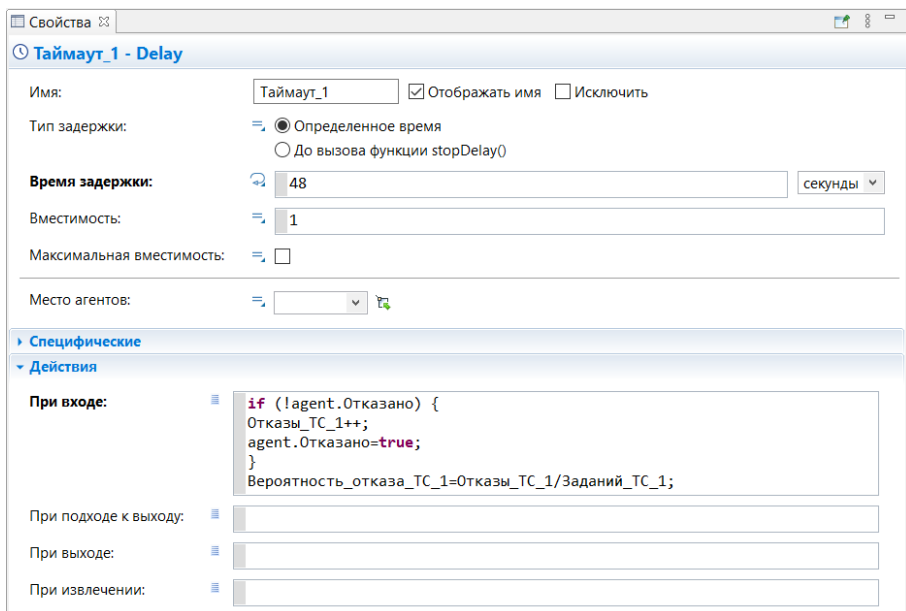


Рисунок 17 – Свойства блока Таймаут

6.2. Агент «Zadanie»

Для проведения имитационных экспериментов в рамках модели также был создан класс агента под названием «Zadanie», реализующий поведение транзакции в системе и хранящий внутри себя следующие параметры:

- Номер_ТС – номер терминальной станции, инициализировавшей данную транзакцию;
- Строки_кода – количество отдельных строк кода, составляющих одно задание;
- Таймер – параметр, хранящий промежуточные значения времени, необходимые для сбора статистики;
- Время_внутри_ТС – время нахождения агента на терминальной станции;
- Время_в_системе – время от начала ввода оператором ТС задания до конца анализа результата обработки задания сервером;

- Время_на_сервере – время обработки задания сервером;
- Отказано – логический параметр, содержащий информацию о том, имел ли место отказ в обслуживании.

На рисунке 18 представлено содержимое класса агента «Zadanie».

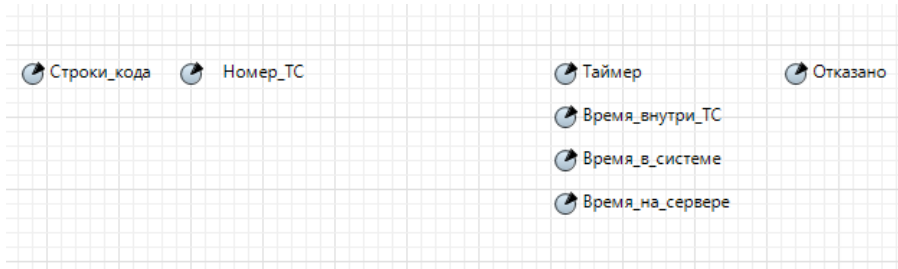


Рисунок 18 – Параметры класса агента «Zadanie»

6.3. Сбор статистики и разделение потоков агентов

В рамках созданной имитационной модели для сбора статистики о времени ожидания ТС начала обслуживания транзакций присутствуют блоки TimeMeasureStart и TimeMeasureStop под названиями Н_Время_Ожидания_1 – Н_Время_Ожидания_5 и К_Время_Ожидания_1 – К_Время_Ожидания_5, на базе данных с которых строятся соответствующие гистограммы.

Аналогично для сбора статистики о времени ожидания операторами ТС результатов выполнения задания присутствуют блоки Н_Время_Обработки_1 – Н_Время_Обработки_5 и К_Время_Обработки_1 – К_Время_Обработки_5.

Для разделения агентов на потоки по признаку номера ТС, в модели присутствуют блоки SelectOutput5 под названиями ТС_селектор, Предсерверный_селектор, Постсерверный_селектор и Селектор_таймаута. Эти блоки производят проверку внутреннего параметра агента Номер_ТС, и на основании имеющегося значения этого параметра производят разделение агентов на потоки там, где это требуется.

Также, блок ТС_Селектор производит сбор статистических данных, таких как:

- время активной работы сервера;
- отношение времени активной работы ко времени простоя для сервера;
- отношение времени активной работы ко времени простоя для каждой ТС;
- количество обработанных сервером заданий;
- эффективность сервера;
- абсолютная и относительная пропускные способности.

На рисунке 19 представлены свойства блока TC_Селектор. Остальные блоки SelectOutput5 отличаются от него отсутствием действий при входе и выходе агентов.

TC_Селектор - SelectOutput5

Имя: ☒ Отображать имя ☐ Исключить

Использовать: ☐ Вероятности ☒ Условия ☐ Номер выхода

Условие 1:

Условие 2:

Условие 3:

Условие 4:

Действия

При входе:

При выходе 1:

При выходе 2:

При выходе 3:

При выходе 4:

При выходе 5:

Рисунок 19 – Свойства блока TC_Селектор

7. Оценка адекватности модели

Для оценки адекватности модели проведём эксперимент с длиной очереди $K = 1$, а также изменим распределение времени выполнения команды $F(\tau_{\text{srv}})$ с $\text{normal}(0.3, 4.8)$ на $\text{normal}(0.6, 9.6)$, тем самым увеличив $M[\tau_{\text{srv}}]$ с 4.8 до 9.6 секунд, т.е в 2 раза, а также дисперсию с 0.3 до 0.6, т.е. также в 2 раза.

При данных параметрах уменьшились такие показатели, как эффективность ТС, производительность ТС (уменьшилась незначительно), и абсолютная пропускная способность, а также отношение времени занятости каждой из ТС ко времени их простоя. Это можно увидеть на диаграммах и гистограммах, представленных на рисунках 20-22.

Однако произошло повышение эффективности сервера и отношение времени занятости сервера к времени его простоя. Это можно объяснить увеличением времени на обработку заданий на сервере. Также увеличилось время ожидания результатов выполнения заданий терминальными станциями.

Основываясь на результатах этого эксперимента можно прийти к выводу, что построенная модель способна обрабатывать отказы в обслуживании, а также адекватно работать при различных конфигурациях. А, следовательно, можно утверждать, что построенная модель адекватна.

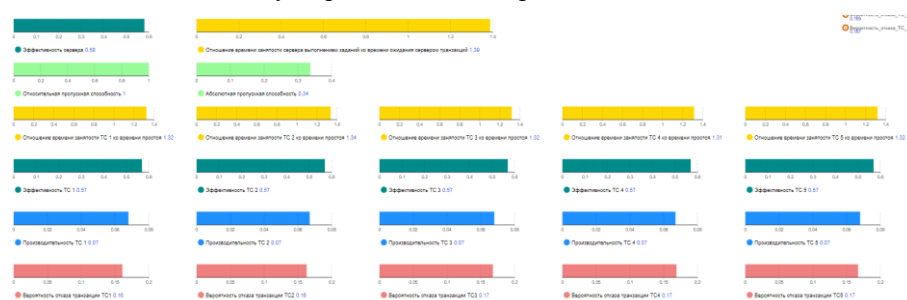


Рисунок 20 – Диаграммы при проверке адекватности

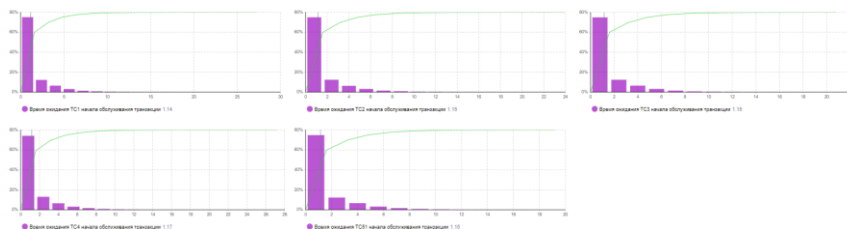


Рисунок 21 – Гистограммы времени ожидания ТС начала обслуживания транзакции сервером при проверке адекватности

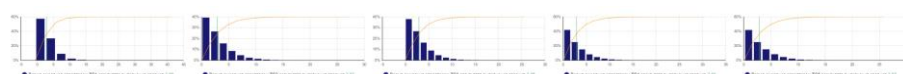


Рисунок 22 – Гистограммы времени ожидания операторами ТС результатов выполнения транзакций сервером при проверке адекватности

8. Оптимизационный эксперимент

В качестве оптимизируемого параметра будем рассматривать длину очереди, которая может принимать дискретные значения от 1 до 4, а также величину стандартного отклонения нормального распределения, определяющего время обработки одной строки кода из задания сервером, которую будем рассматривать на промежутке от 0.1 до 1 с шагом в 0.1.


В качестве критерия оптимальности выберем величину абсолютной пропускной способности: чем она больше, тем более оптимальна работа системы.

Целевой функцией, которую мы будем минимизировать, будет непосредственно абсолютная ПС. Также наложим дополнительное ограничение: время ожидания оператором ТС результатов выполнения задания должно укладываться в диапазон 1-3 минуты, что обеспечивает режим работы on-line. В силу ограничений среды AnyLogic, не позволяющей накладывать в рамках оптимизационного эксперимента больше 7 ограничений, вышеуказанные ограничения наложим только на время ожидания результатов оператором первой ТС. Это допустимо, т.к на большом промежутке времени показатели всех ТС выравниваются.

В оптимизационном эксперименте будем минимизировать целевую функцию на промежутке в 100000 единиц модельного времени.

На рисунке 23 представлены параметры оптимизационного эксперимента. На рисунке 24 – результаты его проведения.

В результате оптимизационного эксперимента, оптимальной оказалась очередь длиной $K = 2$ и стандартным отклонением $\sigma = 0.4$, при которых была достигнута абсолютная ПС 0.389 заданий в минуту.


Оптимизация - Оптимизационный эксперимент

Имя: ☐ Исключить
Агент верхнего уровня:
Целевая функция: ☐ минимизировать ☒ максимизировать

☒ Количество итераций:
☐ Автоматическая остановка
Максимальный размер памяти: МБ

Параметры

Параметры:

Параметр	Тип	Значение			
		Мин.	Макс.	Шаг	Начальное
С_Врем...ростоя	фиксированный				
Вмести...ителя	дискретный	1	4	1	
ТС_1_Вр...ростоя	фиксированный				
ТС_2_Вр...ростоя	фиксированный				
ТС_3_Вр...ростоя	фиксированный				
ТС_4_Вр...ростоя	фиксированный				
ТС_5_Вр...ростоя	фиксированный				
Эффект...ервера	фиксированный				
Абсолютная_ПС	фиксированный				
std	дискретный	0.1	1	0.1	
Ср_Вр...я_TC_1	фиксированный	0			
Ср_Вр...я_TC_2	фиксированный	0			
Ср_Вр...я_TC_3	фиксированный	0			
Ср_Вр...я_TC_4	фиксированный	0			
Ср_Вр...я_TC_5	фиксированный	0			

Модельное время

Остановить:
Начальное время: Конечное время:
Начальная дата: Конечная дата:

Дополнительные условия остановки оптимизации:

Вкл.	Выражение

Ограничения

Требования

Требования (проверяются после "прогона" для определения того, допустимо ли найденное решение):

Вкл.	Выражение	Тип	Гран...
<input checked="" type="checkbox"/>	root.Ср_Вр_Ожидания_TC_1	<=	3.0
<input checked="" type="checkbox"/>	root.Ср_Вр_Ожидания_TC_1	>=	1.0

Рисунок 23 – Параметры оптимизационного эксперимента

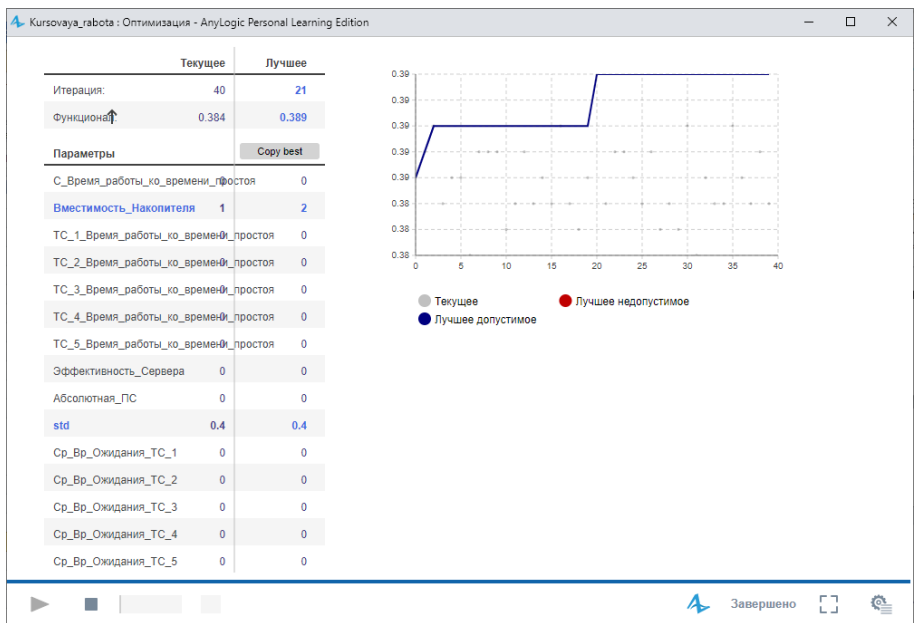


Рисунок 24 – Результаты оптимизационного эксперимента

9. Эксперименты с моделью и статистическая оценка характеристик системы

После получения в оптимизационном эксперименте длины очереди $K=2$ и стандартного отклонения $\sigma = 0.4$, был проведён эксперимент в режиме виртуального времени для сбора данных, необходимых для статистической оценки.

Полученные в результате гистограммы представлены на рисунках 25 и 26. Среднее время ожидания каждой из ТС начала обслуживания транзакций получилось равным 0.25 минут, а среднее время ожидания операторами ТС результатов выполнения транзакций – приблизительно 1.09 минут.

Также в результате проведения эксперимента были получены следующие статистические характеристики системы:

- Значение эффективности каждой из ТС стало равным 0.65;
- Значение эффективности сервера стало равным 0.32;
- Значение производительности каждой из ТС стало равным $0.08 \frac{\text{транзакций}}{\text{мин}}$;
- Значение абсолютной пропускной способности системы стало равным $0.39 \frac{\text{заданий}}{\text{мин}}$;
- Значение относительной пропускной способности стало равным 1;
- Значения вероятностей отказа в обслуживании транзакций, инициализированных каждой из ТС стали равными 0.006;
- Значения отношений времени занятости каждой из ТС ко времени их простоя стали равными 1.83;
- Значение отношения времени занятости сервера выполнением заданий ко времени ожидания сервером транзакций стало равным 0.48.

Диаграммы статистических характеристик системы представлены на

рисунке 27.

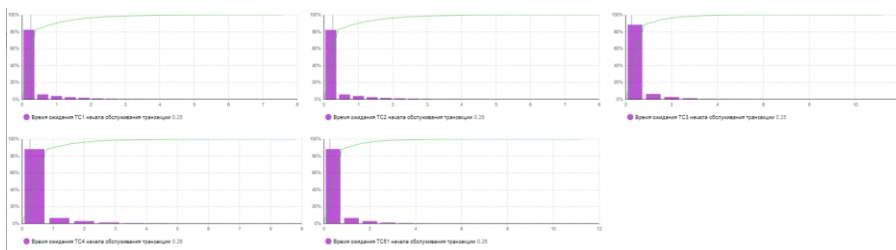


Рисунок 25 – Гистограммы времени ожидания ТС начала обслуживания транзакции сервером

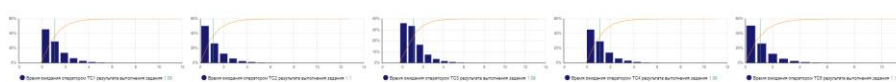


Рисунок 26 – Гистограммы времени ожидания операторами ТС результатов выполнения транзакций сервером



Рисунок 27 – Диаграммы статистических характеристик

10. Итоговые результаты и выводы

В результате выполнения курсовой работы была разработана и создана имитационная модель многотерминальной вычислительной системы с пятью терминальными станциями.

В рамках проведения оптимизационного эксперимента было найдено оптимальное значение длины очереди $K = 2$ и оптимальное стандартное отклонение для нормального распределения, определяющего время обработки сервером одной строки кода из полученного задания $\sigma = 0.4$, при котором достигается наибольшая абсолютная пропускная способность (при данных в варианте задания параметрах модели). При данном значении длины очереди был проведён эксперимент в модельном виртуальном времени, по результатам которого установились следующие статистические характеристики системы:

- Эффективность каждой из ТС $Q_{ТС}^i = 0.65$;
- Эффективность сервера $Q_{сервер} = 0.32$;
- Производительность каждой из ТС $E_{ТС}^i = 0.08 \frac{\text{заданий}}{\text{мин}}$;
- Абсолютная пропускная способность системы $abs = 0.39 \frac{\text{транзакций}}{\text{мин}}$;
- Относительная пропускная способность $rel = 1$
- Вероятности отказа в обслуживании транзакций, инициализированных каждой из ТС $ren^i = 0.006$;
- Отношения времени занятости каждой из ТС ко времени их простоя $T_{ТС}^i = 1.83$;
- Отношение времени занятости сервера выполнением заданий ко времени ожидания сервером транзакций $T_{сервер} = 0.48$.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Стандарт организации. Комплексная система управления качеством деятельности вуза. СТО СГАУ 02068410-004-2007 [Электронный ресурс] // Самарский университет: [сайт]. Общие требования к учебным текстовым документам. – Самара: Изд-во Самарского университета, 2011. – 29 с.
- 2 Порядок выполнения и защиты курсовых работ [Электронный ресурс] // Самарский университет: [сайт]. Метод. указания / сост.: Н.А. Дубровина, А.Г. Лукин, Ю.И. Ряжева. – Самара: Изд-во Самарского университета, 2018. – 34 с.
- 3 Справочная система AnyLogic [Электронный ресурс]. URL: <https://help.anylogic.ru/index.jsp> (дата обращения: 10.06.2021)