Федеральное агентство связи

Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра ВС

Курсовая работа по дисциплине

«Моделирование» на тему:

«Имитационное моделирование вычислительной системы»

Выполнили:

магистранты ф-та ИВТ,

гр. МГ-156

Шлаузер А.И.,

Нелюбин М.А.,

Арчимаев А.С

Проверил:

проф. Родионов А.С.

Новосибирск 2016

СОДЕРЖАНИЕ

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc468648473)

[1 Описание моделируемой системы 4](#_Toc468648474)

[2 Листинг программы моделирования 8](#_Toc468648475)

[3 Результаты моделирования и их анализ 8](#_Toc468648476)

[СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ 9](#_Toc468648477)

# ПОСТАНОВКА ЗАДАЧИ

1. Построить имитационную модель вычислительной системы, соответствующей следующему описанию:

* вычислительная система имеет N ядер и M страниц памяти;
* в вычислительную систему поступают задачи, требующие для выполнения определенное количество ресурсов;
* время поступления задач, количество ресурсов определяется некоторым случайным распределением;
* с некоторой интенсивностью в системе происходит отказ и восстановление ядер.

1. Произвести кодирование модели, провести следующие эксперименты:

* сравнить по среднему времени пребывания задачи в системе 2 режима очереди:

1. FIFO;
2. перескок задачи через очередь по заданному времени ожидания.

* выявить зависимость длины очереди от объема резерва.

# Описание моделируемой системы

Формальное описание моделируемой системы можно представить в виде графа событий, изображенного на рисунке 1.



Рисунок 1 – Граф событий

Последовательную обработку событий осуществляет календарь (монитор) событий. Он извлекает из списка ближайшее по времени событие и производит действия, описанные ниже.

*Постановка задачи в очередь*

Созданная ранее задача добавляется в очередь. Время между приходами задач определяется экспоненциальным распределением.

Событие планируется следующим образом:

Generate(required\_cores\_num, required\_memory\_num, Ti)

Task = <required\_cores\_num, required\_memory\_num>

Schedule(Task, Ti, AddToQueue)

Ниже представлен псевдокод процедуры обработки данного события.

if (resources is available)

{ Schedule(Task, Ti + Twait, BeginCompute) }

*Начало счета задачи*

Событие наступает при условии i – наличии свободных ресурсов, при этом задача удаляется из очереди. Через время планируется событие окончания счета.

Ниже представлен псевдокод процедуры обработки данного события.

cores\_num -= required\_cores\_num

memory\_num -= required\_memory\_num

Schedule(Task, Ti + Twait + Twork, EndCompute)

*Окончание счета задачи*

При наступлении данного события через время планируется новое событие – конец освобождения памяти.

Ниже представлен псевдокод процедуры обработки данного события.

cores\_num += required\_cores\_num

Schedule(Task, Ti + Twait + Twork + Trelease, FreeMemory)

*Конец освобождения памяти*

При наступлении данного события вычисляется время пребывания задачи в системе.

Ниже представлен псевдокод процедуры обработки данного события.

memory\_num += required\_memory\_num

*Отказ ядра*

Отказ ядра возникает со случайным распределением. При этом ядро удаляется из системы. Если ядро было задействовано задачей, задача «убивается» (условие j). При отказе ядра через время планируется событие восстановления ядра.

Событие планируется следующим образом:

Generate(crashed\_core\_id, Ti)

Core = < crashed\_core\_id>

Schedule(Core, Ti, Crash)

Ниже представлен псевдокод процедуры обработки данного события.

working\_cores\_count -=1

crashed\_cores\_count +=1

if (core is not busy) { Cancel(Task, EndCompute) }

Shedule(Core, T+recovery, Recovery)

*Восстановление ядра*

Ядро возвращается в систему.

Ниже представлен псевдокод процедуры обработки данного события.

working\_cores\_count +=1

broken\_cores\_count -=1

Также возможен другой вариант построения модели, в котором задача, находящаяся в очереди, может совершить переход в начало, не дожидаясь, когда отработают остальные. На рисунке 2 изображен граф событий для модели с «перескакиванием». Условием для «перескока» является истечение времени .



Рисунок 2 – Граф событий для модели с «перескакиванием».

# Листинг программы моделирования

# Результаты моделирования и их анализ

Первый эксперимент – максимальное время пребывания в очереди

Таблица 1 – Параметры системы:

|  |  |
| --- | --- |
| Количество ядер: | 64 |
| Количество памяти: | 256 |
| Максимальное количество занимаемых ядер | 16 |
| Максимальное количество занимаемой памяти: | 64 |
| Интенсивность поступления задач: | 2 |
| Интенсивность выполнения задач: | 0,3 |
| Максимальное время работы задачи: | 2 |
| Время освобождения памяти: | 0,0001 \* максимальное время работы задачи |
| Интенсивность отказов: | 0 |
| Интенсивность восстановления: | 0 |
| Время начала поломок: | 25 |

Таблица 2 – Максимальное время пребывания в очереди

|  |  |  |
| --- | --- | --- |
| Время симуляции | FIFO | С перескоком |
| 10 | 0 | 0 |
| 50 | 0 | 0 |
| 100 | 1,954 | 0,68 |
| 250 | 2,487 | 1,244 |
| 500 | 3,65 | 1,764 |
| 750 | 5,874 | 3,704 |
| 1000 | 6,688 | 4,515 |
| 2500 | 6,347 | 5,251 |
| 5000 | 6,584 | 5,798 |
| 7500 | 8,928 | 6,978 |
| 10000 | 9,069 | 7,227 |

Рисунок 1 – Максимальное время пребывания в очереди

# СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ