М. Г. Курносов, А. А. Пазников

ОСНОВЫ ТЕОРИИ ФУНКЦИОНИРОВАНИЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Новосибирск

2015

УДК 004.272
ББК 22.18
К93

Курносов М.Г., Пазников А.А.
К93 Основы теории функционирования распределенных вычислительных систем. – Новосибирск: Автограф, 2015. – 52 с.

ISBN 978-5-9906983-5-2

Практикум содержит описание лабораторных работ по курсу «Теория функционирования распределенных вычислительных систем». Помимо заданий и контрольных вопросов, каждая работа включает теоретический материал, необходимый для ее успешного выполнения.

ББК 22.18

ISBN 978-5-9906983-5-2

Содержание

Основные сокращения

HPC – High-Performance Computing
MPI – Message Passing Interface
ВС – вычислительная система
ВМ – вычислительный модуль
ВЦ – вычислительный центр
ОС – операционная система
ПО – программное обеспечение
ЭВМ – электронная вычислительная машина
ЭМ – элементарная машина
ЭП – элементарный процессор

Описание курса

Курс «Теория функционирования распределенных вычислительных систем» состоит из двух частей, рассчитанных на два учебных семестра. Первая часть посвящена анализу надежности и живучести распределенных вычислительных систем (ВС). Во второй части рассматриваются модели и алгоритмы организации функционирования ВС.

Основной теоретический материал содержится в следующих книгах, которые являются базовыми по курсу:

1. Хорошевский В.Г. *Архитектура вычислительных систем*. – М.: МГТУ им. Н.Э. Баумана, 2008. – 520 с.
2. Евреинов Э.В., Хорошевский В.Г. *Однородные вычислительные системы*. – Новосибирск: Наука. Сибирское отделение, 1978. – 319 с.
3. Хорошевский В.Г. *Инженерный анализ функционирования вычислительных машин и систем*. – М.: Радио и связь, 1987. – 256 с.

В течении каждого учебного семестра подразумевается проведение лекционных, лабораторных (практических) занятий, контрольных работ и экзамена/зачета.

Оценка за экзамен или получение зачета зависит от результата выполнения лабораторных работ.

Каждая лабораторная работа (задание) оценивается определенным числом баллов и имеет *крайний срок сдачи* (deadline). На выполнение и сдачу каждого задания отводится фиксированное число k занятия. Каждое k-ое занятие – это крайний срок сдачи соответствующей лабораторной работы.

Если задание не сдано в срок, за него выставляется минимальное количество баллов. Количество набранных баллов влияет на максимальную оценку за курс. Для допуска к экзамену все задания должны быть выполнены и сданы. Во время сессии задания не принимаются.

Оценка за экзамен определяется числом набранных баллов в процентах от максимально возможного:

- 90–100% от максимального числа баллов – оценка «отлично»;
- 80–89% – «хорошо»;
- 70–79% – «удовлетворительно»;
- $<$ 70% – «неудовлетворительно».

1. Порядок выполнения лабораторных работ

1.1. Этапы выполнения лабораторной работы

Рекомендуется придерживаться следующего порядка выполнения лабораторных работ.

1. Ознакомиться с заданием на лабораторную работу.

2. Изучить необходимый теоретический материал. Здесь следует использовать лекций и другие рекомендуемые в практикуме источники.

3. Написать программу, решающую поставленную в лабораторной работе задачу. Проверить корректность программы.

4. В соответствии с заданием провести эксперименты (построить графики, заполнить таблицы и т. д.).

5. Выполнить анализ корректности полученных в ходе экспериментов результатов. Например, вид и поведение кривых на графиках; допустимость числовых значений в таблицах и пр.

6. Ответить на контрольные вопросы.

1.2. Защита лабораторной работы

Защита лабораторной работы проходит в три этапа.

1. Проверка исходного кода программы. Здесь требуется обосновать выбранные способы реализации программы. Если в ходе проверки установлено, что код программы разработан другим лицом, лабораторная работа не принимается.

2. Проверка результатов экспериментов и отчета. Демонстрируются построенные графики, таблицы. Обосновываются поведение кривых (возрастание, убывание, экстремумы) и числовые значения в таблицах.

3. Ответы на контрольные вопросы.

1.3. Требования к программам

1. Программы разрабатываются для операционной системы GNU/Linux на любом языке программирования (например, C/C++, C#, Java, Python).

2. Программы должны собираться без ошибок и предупреждений. Для компилятора GCC (C/C++) рекомендуется использовать опции `-O2 -Wall`.

3. Оформление исходного кода программы должно соответствовать принятым в рамках курса соглашениям или аналогичным:

– K&R style;
– GNU coding standards;
– Google C++ style guide;
– Linux kernel coding style.

4. Программный код должен быть разработан самостоятельно. Решения, полностью заимствованные из сети Интернет, и «работы-близнецы» к защите не допускаются. В случае частичных заимствований исходного года, это должно быть обозначено комментариями в программе и отражено в отчете. Желательно привести ссылки.

Оформление отчета

Основной текст отчета оформляется одинаковым шрифтом, например Times New Roman, кегль – 12 пт. Межстрочный интервал – одинарный. Отступ первой строки каждого абзаца – 1 см. Выравнивание содержимого основных абзацев текста – «по ширине », заголовки разделов выравниваются «по центру» или левому краю.

Исходный код программ и результаты вывода на экран оформляются моноширинным шрифтом, например: Courier New, Consolas.

Рисунки, графики и таблицы должны быть выровнены по центру страницы и иметь подписи.

Формат отчета PDF. Для его подготовки рекомендуется использовать пакеты LibreOffice, LaTeX, Google Docs. Графики желательно оформлять средствами gnuplot, LibreOffice, Asymptote, MetaPost или R.

Отчет должен содержать нижеследующие части.

1. **Титульный лист**. Он является первым листом и не нумеруется. Поля титульного листа должны быть выдержаны в тех же размерах, что и вся работа. Выравнивание содержимого всех строк титульного листа «по центру». Кроме строки «Выполнил», ее выравнивание – по правому краю или с фиксированным отступом от левого края. Шрифт – Times New Roman, кегль – 12 пт. В шапке титульного листа указывается: ведомственная принадлежность учебного заведения, название учебного заведения, название кафедры, на которой читается дисциплина. В центре титульного листа приводится название работы. Оно должно быть выделено на фоне остального текста: посредством полужирного шрифта, либо посредством прописных (заглавных) букв. Ниже следует строка «Выполнил», под которой указы-

вается автор и номер его учебной группы. Внизу титульного листа приводятся город, в котором расположено учебное заведение, и год выполнения работы.

2. **Описание задания**. В этом разделе формулируется задание на лабораторную работу.

3. **Результаты выполнения работы**. Здесь приводятся полученные в ходе выполнения работы результаты: графики, таблицы, схемы (по заданию).

Результаты экспериментов должны сопровождаться описанием условий их проведения:

– вычислительная система: модель процессора, объем оперативной памяти;
– системное ПО: версия операционной системы, версия компилятора, ключи компиляции программы;
– входные данные: значения параметров модели и параметры запуска тестов.

Степень детализации описания условий проведения экспериментов, должна быть такой, чтобы посторонний специалист мог повторить эксперименты.

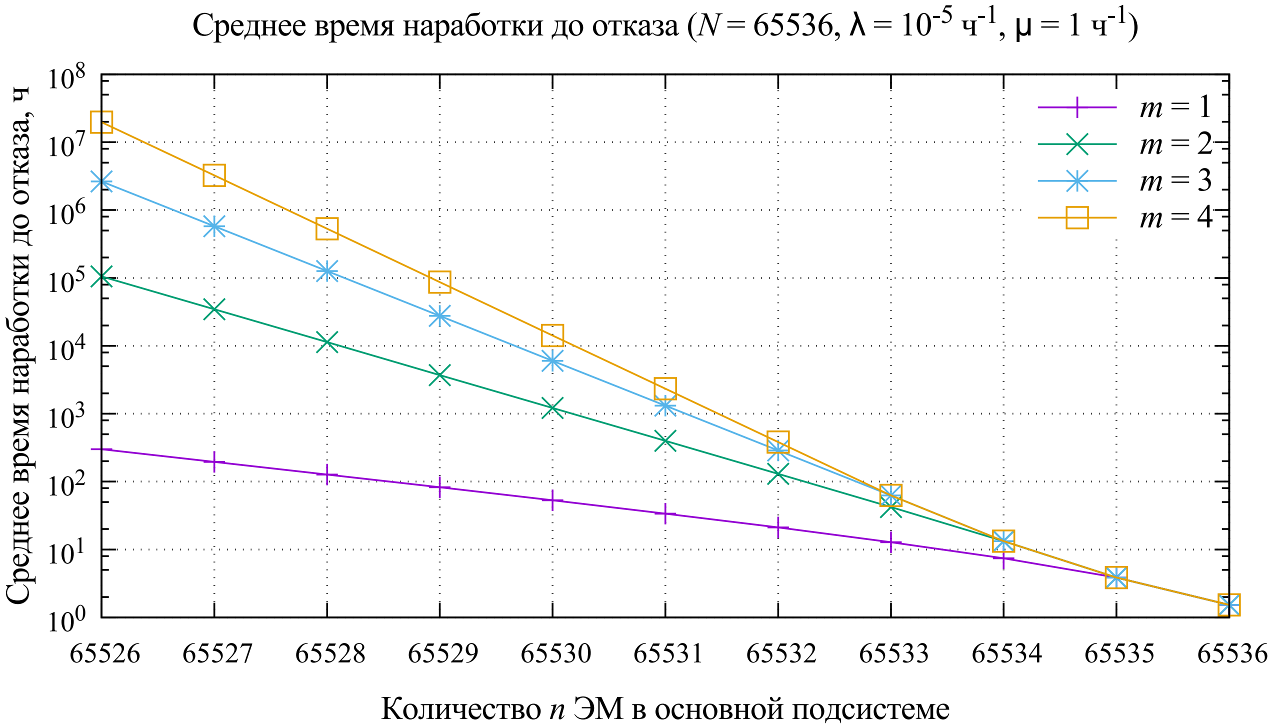

Рис. 1.1. Зависимость среднего времени Θ наработки до отказа от числа n ЭМ в основной подсистеме: $N = 65536$, $\lambda = 10^{-5}$ ч$^{-1}$, $\mu = 1$ ч$^{-1}$.

Каждый график в отчете должен быть подписан. Следует указать зависимость чего отчего на нем приведена. Оси графиков должны содержать подписи и единицы измерения. Например:

– время выполнения программы, с;
– среднее время Θ безотказной работы ВС, ч;
– количество N ЭМ;
– значение целевой функции $T(S)$;

Если на графике несколько кривых, следует пронумеровать их, а в подрисуночной подписи дать описания каждой из них (см. рис. 1.1).

4. **Список использованных источников**. На все источники должны быть даны ссылки в тексте отчета.

2. Надежность и живучесть ВС

2.1. Среднее время безотказной работы и восстановления ВС

2.1.1. Модель функционирования ВС со структурной избыточностью

Имеется распределенная ВС, укомплектованная N одинаковыми элементарными машинами. Основная подсистема (вычислительное ядро) ВС состоит их n ЭМ, $N-n$ элементарных машин составляют структурную избыточность. Заданы λ – интенсивность потока отказов любой из N элементарных машин ($[\lambda] = 1/\text{ч}$), m – количество восстанавливающих устройств восстанавливающей системы и μ – интенсивность потока восстановления элементарных машин одним восстанавливающим устройством ($[\mu] = 1/\text{ч}$).

В инженерной практике при анализе надежности ВС наиболее употребительны такие показатели, как *математическое ожидание* Θ *времени безотказной работы* (средняя наработка до отказа) и *среднее время* T *восстановления* ВС, которые равны [1, С. 419]:

$$\Theta = \int\limits_0^\infty R(t)dt, \tag{2.1}$$

$$T = \int\limits_0^\infty t dU(t), \tag{2.2}$$

где $R(t)$ – функция надежности ВС, а $U(t)$ – функция восстановимости системы.

Для распределенных ВС показатели Θ и T можно рассчитывать «частотным» методом [1, С. 427], который обеспечивает результаты, хорошо согласующиеся с более точными вычислениями:

$$\Theta = \begin{cases} \sum\limits_{j=n+1}^{N} \frac{1}{j\lambda} \prod\limits_{l=n}^{j-1} \frac{\mu_l}{l\lambda} + \frac{1}{n\lambda}, & \text{при } n \neq N, \\ \frac{1}{N\lambda}, & \text{при } n = N. \end{cases} \tag{2.3}$$

$$T = \begin{cases} \frac{1}{\mu_0} \prod\limits_{l=1}^{n-1} \frac{l\lambda}{\mu_l} + \sum\limits_{j=1}^{n-1} \frac{1}{j\lambda} \prod\limits_{l=j}^{n-1} \frac{l\lambda}{\mu_l}, & \text{при } n > 1, \\ \frac{1}{\mu_0}, & \text{при } n = 1. \end{cases} \tag{2.4}$$

2.1.2. Задание

1. **Написать программу** расчета частотным методом математического ожидания Θ времени безотказной работы и среднего времени T восстановления ВС со структурной избыточностью.

2. **Построить графики зависимости значений показателя** Θ от параметров λ, μ, m и n.

2.1. Построить график зависимости $\Theta(n)$. Использовать следующие значения параметров:

$$N = 65536, \quad \lambda = 10^{-5}, \quad m = 1,$$

$$n = 65527, 65528, \ldots, 65536; \quad \mu \in \{1, 10, 100, 1000\}.$$

2.2. Построить график зависимости $\Theta(n)$. Параметры:

$$N = 65536, \quad \mu = 1, \quad m = 1,$$

$$n = 65527, 65528, \ldots, 65536; \quad \lambda \in \{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}\}.$$

2.3. Построить график зависимости $\Theta(n)$. Параметры:

$$N = 65536, \quad \mu = 1, \quad \lambda = 10^{-5},$$

$$n = 65527, 65528, \ldots, 65536; \quad m \in \{1, 2, 3, 4\}.$$

3. **Построить графики зависимости значений показателя** T от параметров λ, μ, m и n.

3.1. Построить график зависимости $T(n)$. Параметры:

$$N = 1000, \quad \lambda = 10^{-3}, \quad m = 1,$$

$$n = 900, 910, \ldots, 1000; \quad \mu \in \{1, 2, 4, 6\}.$$

3.2. Построить график зависимости $T(n)$. Параметры:

$$N = 8192, \quad \mu = 1, \quad m = 1,$$

$$n = 8092, 8102, \ldots, 8192; \quad \lambda \in \{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}\}.$$

3.3. Построить график зависимости $T(n)$. Параметры:

$$N = 8192, \quad \mu = 1, \quad \lambda = 10^{-5},$$

$$n = 8092, 8102, \ldots, 8192; \quad m \in \{1, 2, 3, 4\}.$$

2.1.3. Контрольные вопросы

1. Дать определение ВС со структурной избыточностью, переходного и стационарного режимов функционирования ВС.

2. Описать стохастическую модель функционирования ВС со структурной избыточностью (трехпараметрическая модель: λ, μ, ν).

3. Дать определение основных показателей надежности ВС в переходном режиме функционирования.

4. Дать определение основных показателей надежности ВС в стационарном режиме функционирования.

5. Объяснить методику вывода расчетных формул для показателей надежности в переходном и стационарном режимах.

6. По построенным графикам определить, какое количество ЭМ достаточно иметь в резерве для обеспечения среднего времени наработки до отказа ВС не ниже времени наработки до отказа одной ЭМ.

7. Варьирование каких параметров позволяет увеличить значение показателя Θ и уменьшить значение T?

2.2. Надежность ВС в стационарном режиме функционирования

2.2.1. Стационарный режим функционирования ВС со структурной избыточностью

Имеется распределенная ВС со структурной избыточностью, укомплектованная N одинаковыми элементарными машинами. Основная подсистема (вычислительное ядро) ВС состоит их n ЭМ, $N-n$ элементарных машин составляют структурную избыточность. Заданы λ – интенсивность потока отказов любой из N элементарных машин ($[\lambda] = 1/\text{ч}$), m – количество восстанавливающих устройств восстанавливающей системы и μ – интенсивность потока восстановления элементарных машин одним восстанавливающим устройством ($[\mu] = 1/\text{ч}$).

При анализе надежности ВС в *стационарном режиме* работы используются такие показатели, как *функция $R^*(t)$ оперативной надежности*, *функция $U^*(t)$ оперативной восстановимости* и *коэффициент S готовности* ВС. Известно [1, С. 431], что функцию $R^*(t)$ можно рассчитать по формуле

$$R^*(t) = \sum_{i=n}^{N} P_i Q_i(t), \tag{2.5}$$

где $Q_i(t) = \sum\limits_{l=0}^{\infty} u_l(t) \sum\limits_{r=0}^{i-n+l} \pi_r(t)$, а значения $r, l \in E_0^{\infty} = \{0, 1, 2, \ldots\}$.

Функции $u_l(t)$ и $\pi_r(t)$ имеют следующий вид:

$$\pi_r(t) = \frac{(i\lambda t)^r}{r!} e^{-i\lambda t},$$

$$u_l(t) = \frac{(\mu t)^l}{l!} \left[\Delta(N-i-m) m^l e^{-m\mu t} + \Delta(m-N+i)(N-i)^l e^{-(N-i)\mu t}\right],$$

$$\Delta(x) = \begin{cases} 1, & \text{если } x \geq 0, \\ 0, & \text{если } x < 0, \end{cases}$$

и считается, что $0^0 = 1$.

В [1, С. 433] приведены формулы расчета вероятностей P_i состояний ВС для стационарного режима работы. Так, для случая $m = 1$, имеем

$$P_j = \left(\frac{\mu}{\lambda}\right)^j \frac{1}{j!} \left[\sum_{l=0}^{N} \left(\frac{\mu}{\lambda}\right)^l \frac{1}{l!}\right]^{-1}, \quad j \in E_0^N. \tag{2.6}$$

2.2.2. Задание

1. **Написать программу** расчета функции оперативной $R^*(t)$ надежности, функции $U^*(t)$ оперативной восстановимости и коэффициента S готовности ВС со структурной избыточностью. Для приближенного вычисления факториала больших чисел рекомендуется использовать формулу Стирлинга (см. приложение).

2. **Построить график** зависимости функции $R^*(t)$ оперативной надежности для следующих значений параметров модели:

$$N = 10, \quad n \in \{8, 9, 10\}, \quad m = 1,$$

$$\lambda = 0.024 \text{ 1/ч}, \quad \mu = 0.71 \text{ 1/ч}, \quad t = 0, 2, 4, \ldots, 24 \text{ ч}.$$

3. **Построить график** зависимости функции $U^*(t)$ оперативной восстановимости для следующих значений параметров:

$$N = 16, \quad n \in \{10, 11, \ldots, 16\}, \quad m = 1,$$

$$\lambda = 0.024 \text{ 1/ч}, \quad \mu = 0.71 \text{ 1/ч}, \quad t = 0, 2, 4, \ldots, 24 \text{ ч}.$$

4. Заполнить таблицу 2.1 значений показателя S для следующих значений параметров:

$$N = 16, \quad \lambda = 0.024 \, 1/\text{ч}, \quad \mu = 0.71 \, 1/\text{ч}.$$

Таблица 2.1. Значения коэффициента S готовности ВС

n	m	
	1	16
11		
12		
13		
14		
15		
16		

2.2.3. Контрольные вопросы

1. Дать определение ВС со структурной избыточностью.

2. Описать стохастическую модель функционирования ВС со структурной избыточностью (трехпараметрическая модель: λ, μ, ν).

3. Пояснить смысл вероятностей состояния системы в стационарном режиме.

4. Дать определение основных показателей надежности ВС в стационарном режиме функционирования.

5. Объяснить суть методики вывода формул для расчета показателей надежности в стационарном режиме.

6. Варьирование каких параметров позволяет увеличить (уменьшить) значение функций $R^*(t)$, $U^*(t)$ и коэффициента S?

2.3. Переходный режим функционирования живучих ВС

2.3.1. Модель функционирования живучей ВС

Имеется живучая распределенная вычислительная система, укомплектованная N одинаковыми элементарными машинами. Заданы минимально допустимое число n работоспособных ЭМ, λ – интенсивность потока отказов любой из N элементарных машин ($[\lambda] = 1/\text{ч}$), m – количество восстанавливающих устройств восстанавливающей системы и μ – интенсивность потока восстановления элементарных машин одним восстанавливающим устройством ($[\mu] = 1/\text{ч}$).

В инженерной практике при анализе функционирования живучих ВС рассматривают *вектор* Θ *среднего времени безотказной работы* и *вектор* T *среднего времени восстановления* системы [1, С. 465]:

$$\mathbf{\Theta} = (\Theta_n, \Theta_{n+1}, \ldots, \Theta_N),$$

$$\mathbf{T} = (T_n, T_{n+1}, \ldots, T_N).$$

Для расчета компонент векторов рекомендуется использовать «частотный метод» (формулы 2.3 и 2.4).

2.3.2. Задание

1. **Написать программу** расчета частотным методом компонентов вектора $\mathbf{\Theta}$ среднего времени безотказной работы и вектора $\mathbf{T}$ среднего времени восстановления живучей вычислительной системы.

2. **Рассчитать и занести в таблицу** значения векторов $\mathbf{\Theta}$ и $\mathbf{T}$. По таблице построить графики, отражающие зависимость значений компонентов $\mathbf{\Theta}$ и $\mathbf{T}$ от значений параметров λ, μ, m и n. Параметры модели:

$$N = 65536, \quad \lambda \in \{10^{-6}, 10^{-7}, 10^{-5}\}, \quad \mu \in \{1, 10, 100, 1000\},$$

$$m \in \{1, 2, 3\}, \quad n \in \{65527, 65528, \ldots, 65536\}.$$

Таблица 2.2. Значения векторов $\mathbf{\Theta}$ и $\mathbf{T}$

№	λ	μ	m	n	$\mathbf{\Theta} = (\Theta_n, \Theta_{n+1}, \ldots, \Theta_N)$	$\mathbf{T} = (T_n, T_{n+1}, \ldots, T_N)$

2.3.3. Контрольные вопросы

1. Дать определения живучей ВС. Объяснить отличие структурной живучести ВС от потенциальной.

2. Сравнить функции производительности ВС со структурной избыточностью и живучей ВС.

3. Дать определение основных показателей живучести ВС.

4. Перечислить параметры, варьирование которых позволяет изменять значения компонентов векторов $\mathbf{\Theta}$ и $\mathbf{T}$ в большую/меньшую сторону.

2.4. Континуальный подход к анализу живучих ВС

2.4.1. Модель функционирования живучей ВС

Рассмотрим живучую ВС, состоящую из N элементарных машин и m восстанавливающих устройств. Система находится в состоянии $k \in E_0^N$, $E_0^N = \{0, 1, \ldots, N\}$. Качество функционирования такой системы оценивается *функциями потенциальной живучести* $N(i,t)$ и *занятости восстанавливающей системы* $M(i,t)$ [1, 464]. Функции $N(i,t)$ и $M(i,t)$ характеризуют в момент времени $t \geq 0$ среднюю производительность ВС и среднюю загруженность восстанавливающей системы, если ВС начала функционировать с $i \in E_0^N$ работоспособными ЭМ.

Обозначим через $n(i,t)$ среднее число работоспособных машин в момент $t \geq 0$, при условии, что система начала функционировать в состоянии $i \in E_0^N$. Тогда функция потенциальной живучести имеет вид

$$N(i,t) = n(i,t)/N.$$

Функция занятости восстанавливающей системы:

$$M(i,t) = m(i,t)/N,$$

где $m(i,t)$ – это математическое ожидание числа занятых восстанавливающих устройств в момент времени $t \geq 0$, при условии, что система начала функционировать в состоянии $i \in E_0^N$.

Для расчета функций $n(i,t)$ и $m(i,t)$ получены формулы для различных случаев производительности восстанавливающей системы [1, С. 471].

Случай 1. Восстанавливающая система имеет высокую производительность: для любого $t \geq 0$ выполняется условие

$$N - n(i,t) \leq m, \tag{2.7}$$

где $i \in E_{N-m}^N = \{N-m, N-m+1, \ldots, N\}$.

Последнее неравенство выполняется на промежутке времени $[0, \infty)$, если

$$N\lambda \leq m(\lambda + \mu).$$

Тогда справедливы формулы

$$N(i,t) = \frac{\mu}{\lambda + \mu} + \frac{i\lambda - (N-i)\mu}{N(\lambda + \mu)} e^{-(\lambda+\mu)t},$$

$$M(i,t) = \frac{N\lambda}{m(\lambda+\mu)} - \frac{i\lambda - (N-i)\mu}{m(\lambda+\mu)} e^{-(\lambda+\mu)t}.$$

Для стационарного режима имеют место формулы для расчета коэффициентов потенциальной живучести ВС и занятости восстанавливающей системы:

$$N = \mu/(\lambda+\mu),$$

$$M = N\lambda/(m(\lambda+\mu)).$$

Случай 2. Восстанавливающая система имеет невысокую производительность: при любом $t \geq 0$

$$N - n(i,t) > m, \quad i \in E_0^{N-m-1} = \{0, 1, \ldots, N-m-1\}. \tag{2.8}$$

В этом случае функция и коэффициент потенциальной живучести ВС имеют вид:

$$N(i,t) = \frac{m\mu}{N\lambda} + \frac{i\lambda - m\mu}{N\lambda} e^{-\lambda t}, \quad i \in E_0^{N-m-1},$$

$$N = \frac{m\mu}{N\lambda}.$$

Функция занятости восстанавливающей системы тождественно равна константе: $M(i,t) = M = 1$.

Случай 3. Восстанавливающая система имеет невысокую производительность, но $n(i,0) = i$, $i \in E_{N-m}^{N}$.

В этом случае до момента времени t^*, когда впервые нарушится условие (2.7), будут справедливы формулы для случая 1. С момента времени t^* будут справедливы формулы для случая 2, в которых следует положить $i = N - m - 1$.

Случай 4. Восстанавливающая система имеет высокую производительность, однако $n(i,0) = i$, $i \in E_0^{N-m-1}$.

В этом случае вначале будут справедливы формулы для случая 2; с момента t^*, когда впервые нарушается условие (2.8), справедливы формулы для случая 1.

2.4.2. Задание

1. **Написать программу** расчета функций $N(i,t)$ потенциальной живучести ВС и $M(i,t)$ занятости восстанавливающей системы.

Построить графики функций $N(i,t)$, $M(i,t)$ для конфигурации живучей ВС, восстанавливающая система которой имеет высокую производительность. Построить следующие графики (две кривые на одном графике):

– графики функции $N(i,t)$ для двух различных значений λ;
– графики функции $N(i,t)$ для двух различных значений μ;

– графики функции $N(i,t)$ для двух различных значений m;
– графики функции $M(i,t)$ для двух различных значений λ;
– графики функции $M(i,t)$ для двух различных значений μ;
– графики функции $M(i,t)$ для двух различных значений m.

2. **Рассчитать значения коэффициентов** N, M для конфигурации живучей ВС, восстанавливающая система которой имеет высокую производительность. Результаты расчетов оформить в виде таблицы 2.3. Рекомендуемые входные данные:

– $N = 100$;
– $\lambda \in \{0.025, 0.25\}$;
– $\mu \in \{0.6, 0.8\}$;
– $m \in \{1, 2, 3\}$;
– $n = 0.9N$;
– $i = 0.9N + 1$.

Таблица 2.3. Значения коэффициентов N и M

№	λ	μ	m	N	M

2.4.3. Контрольные вопросы

1. Дать определения живучей ВС.

2. Описать модель функционирования живучей. Описать задачи реконфигуратора.

3. Пояснить суть континуального подхода к расчету показателей живучести ВС.

4. Описать дискретный подход к расчету функции $n(i,t)$.

5. Сформулировать условие высокой производительности восстанавливающей системы.

2.5. Технико-экономическая эффективность функционирования ВС

2.5.1. Эксплуатационные расходы функционирования ВС

Пусть имеются вычислительная и восстанавливающая системы, состоящие соответственно из N элементарных машин и m восстанавливающих устройств, $N \geq m \geq 0$, λ и μ – интенсивности отказов ЭМ и восстановления отказавших ЭМ одним ВУ; c_1 и c_2 – стоимости эксплуатации одной ЭМ и содержания одного ВУ в единицу времени.

Расходы, вызванные простоями ЭМ из-за отказов и простоями ВУ из-за недостатка отказавших ЭМ, будем называть бесполезными. Для оценки этих расходов используют функцию $\Gamma(i,t)$ *математического ожидания бесполезных эксплуатационных расходов* к моменту времени $t \geq 0$ при условии, что система находилась в состоянии $i \in E_0^N = \{0, 1, \ldots, N\}$ при $t = 0$ (т.е. в системе было i работоспособных ЭМ в момент начала функционирования ВС). Для вычисления функции $\Gamma(i,t)$ рассмотрим два случая функционирования восстанавливающей системы.

Случай 1. Восстанавливающая система имеет высокую производительность, т.е. выполняется неравенство

$$N\lambda \leq m\mu.$$

Тогда математическое ожидание $\Gamma(i,t)$ бесполезных эксплуатационных расходов рассчитывается по формуле

$$\Gamma(i,t) = -\varepsilon_i + \gamma t + \varepsilon_i \delta(t), \quad i \in E_{N-m}^N,$$

$$\delta(t) = e^{-(\lambda+\mu)t},$$

$$\gamma = \frac{N\lambda}{\lambda+\mu}(c_1 - c_2) + mc_2,$$

$$\varepsilon_i = \frac{i\lambda - (N-i)\mu}{(\lambda+\mu)^2}(c_1 - c_2).$$

Случай 2. Восстанавливающая система имеет низкую производительность:

$$N\lambda > m\mu.$$

В этом случае

$$\Gamma(i,t) = -\varepsilon_i + \gamma t + \varepsilon_i \delta(t), \quad i \in E_{N-m}^N,$$

$$\delta(t) = e^{-\lambda t},$$

$$\gamma = \frac{N\lambda - m\mu}{\lambda}c_1,$$

$$\varepsilon_i = \frac{i\lambda - m\mu}{\lambda^2} c_1.$$

При длительной эксплуатации ВС средние бесполезные эксплуатационные расходы не зависят от начального состояния и вычисляется по формуле:

$$\Gamma(t) = \gamma t.$$

2.5.2. Задание

1. **Написать программу** расчета функции $\Gamma(i,t)$ математического ожидания бесполезных эксплуатационных расходов.

Построить графики функций $\Gamma(i,t)$ для случая, когда восстанавливающая система имеет высокую производительность, и для случая, восстанавливающая система имеет низкую производительность. Построить следующие графики (две кривые):

– графики функции $\Gamma(i,t)$ для двух различных значений λ;
– графики функции $\Gamma(i,t)$ для двух различных значений μ;
– графики функции $\Gamma(i,t)$ для двух различных значений c_1;
– графики функции $\Gamma(i,t)$ для двух различных значений c_2.

2. **Написать программу** для вычисления функции $\Gamma(i,t)$ математического ожидания бесполезных эксплуатационных расходов в стационарном режиме и построить графики функции для всех значений параметров модели из пункта 1 задания.

Рекомендуемые входные данные:

– $N = 100$;
– $\lambda \in \{0.025, 0.25\}$;
– $\mu \in \{0.6, 0.8\}$;
– $m \in \{1, 2, 3\}$;
– $i = 0.9N + 1$;
– $c_1 \in \{1, 2\}$;
– $c_2 \in \{0.5, 1\}$.

2.5.3. Контрольные вопросы

1. Что такое цена быстродействия вычислительных систем?

2. Сформулировать закон Гроша.

3. Оцените цену одной операции в секунду для вычислительной машины, на которой выполнялась лабораторная работа.

4. Дать определения основных показателей технико-экономической эффективности ВС.

5. Описать суть методики расчета математического ожидания $\Gamma(i, t)$ бесполезных эксплуатационных расходов ВС.

6. Изменение каких параметров ВС уменьшает бесполезные эксплуатационные расходы и увеличивают доход от эксплуатации вычислительных систем?

3. Организация функционирования ВС

3.1. Планирование решения задач на ВС

3.1.1. Задача двумерной упаковки прямоугольников

Имеется распределенная вычислительная система, укомплектованная n элементарными машинами. Задан набор из m параллельных задач (task, job). Каждая задача $j \in J = \{1, 2, \ldots, m\}$ характеризуется временем t_j решения и количеством r_j элементарных машин, необходимых для нее.

Требуется построить *расписание* (schedule) S решения параллельных задач на распределенной ВС. Для каждой задачи необходимо определить момент времени $\tau_j \in \mathbb{R}$ начала решения ее ветвей и их распределение по элементарным машинам.

Пусть $x_{ji} \in C = \{1, 2, \ldots, n\}$ – номер ЭМ, на которую распределена ветвь $i \in \{1, 2, \ldots, r_j\}$ задачи $j \in J$. Обозначим через

$$J(t) = \{j \in J \,|\, \tau_j \le t \le \tau_j + t_j\}$$

множество задач, решаемых на распределенной ВС в момент времени t.

Расписание

$$S = (\tau_1, \tau_2, \ldots, \tau_m; x_{11}, x_{12}, \ldots, x_{1r_1}, \ldots, x_{m1}, x_{m2}, \ldots, x_{mr_m})$$

будем называть допустимым, если оно удовлетворяет следующим ограничениям.

1. В любой момент времени на ресурсах распределенной ВС решается не более n ветвей параллельных задач:

$$\sum_{j \in J(t)} r_j \le n, \quad \forall t \in \mathbb{R}.$$

2. В любой момент времени $t \in \mathbb{R}$ ветви параллельных задач решаются на разных элементарных машинах:

$$\prod_{j \in J(t)} \prod_{j' \in J(t) \setminus \{j\}} (x_{ji} - x_{j'i'}) \ne 0, \quad i = 1, 2, \ldots, r_j, \quad i' = 1, 2, \ldots, r'_j.$$

Обозначим через Ω множество допустимых расписаний. В качестве показателя оптимальности расписаний будем использовать время $T(S)$ окончания решения последней задачи

$$T(S) = \max_{j \in J}\{\tau_j + t_j\}.$$

Итак, требуется найти допустимое расписание S, доставляющее минимум целевой функции $T(S)$. Формально

$$T(S) = \max_{j \in J}\{\tau_j + t_j\} \to \min_{S \in \Omega}, \tag{3.1}$$

при ограничениях:

$$\sum_{j \in J(t)} r_j \le n, \quad \forall t \in \mathbb{R}, \tag{3.2}$$

$$\prod_{j \in J(t)} \prod_{j' \in J(t) \setminus \{j\}} (x_{ji} - x_{j'i'}) \ne 0, \quad i = 1, 2, \ldots, r_j, \quad i' = 1, 2, \ldots, r'_j, \tag{3.3}$$

$$x_{ji} \in C, \tau_j \in \mathbb{R}. \tag{3.4}$$

Задача (3.1)–(3.4) относится к дискретной оптимизации и является трудноразрешимой.

Один из подходов к приближенному решению задачи основан на ее сведении к задаче двумерной упаковки прямоугольников в полуограниченную полосу (2D strip packing, 2DSP). Каждая параллельная программа $j \in J$ представляется в виде прямоугольника шириной r_j и высотой t_j условных единиц. Ширина полосы составляет n условных единиц, высота – не ограничена. Требуется упаковать прямоугольники в полосу без их вращений и пересечений так, чтобы высота упаковки была минимальной.

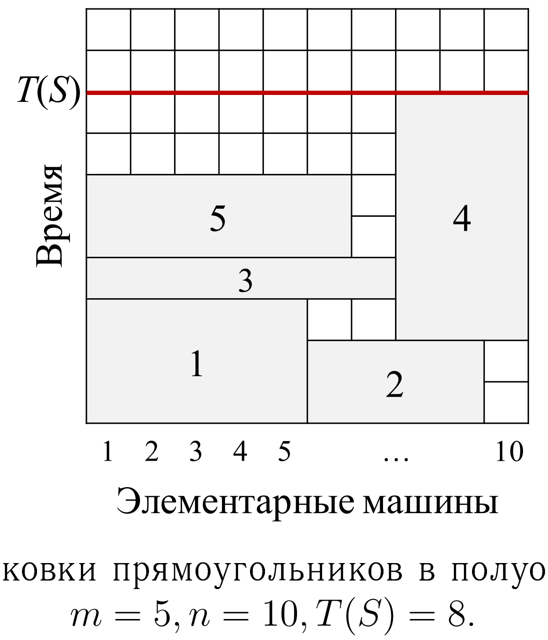

Рис. 3.1. Пример упаковки прямоугольников в полуограниченную полосу: $m = 5, n = 10, T(S) = 8$.

На рис. 3.1 приведен пример упаковки 5 прямоугольников (задач) в полосу шириной 10 условных единиц (элементарных машин). Задача с номером 1 запускается на решение в нулевой момент времени ($\tau_1 = 0$) и использует элементарные машины 1–5 ($x_{11} = 1$, $x_{12} = 2$, …, $x_{15} = 5$), задача 4 начинает решаться в момент времени 2 и использует ЭМ 8, 9 и 10. Значение целевой функции $T(S) = 8$.

3.1.2. Задание

1. **Написать программу**, реализующую алгоритмы NFDH – Next-Fit Decreasing Height и FFDH – First-Fit Decreasing Height для приближенного решения задачи (3.1)–(3.4).

В качестве входных параметров программа получает имя файла с набором задач, количество n ЭМ в системе и название алгоритма (NFDH или FFDH). Результат работы программы:

- расписание S решения задач;
- значение $T(S)$ целевой функции;
- отклонение ε значения целевой функции от ее нижней границы T';
- время t выполнения программы в секундах.

$$\varepsilon = \frac{T(S) - T'}{T'}, \quad T' = \frac{1}{n}\sum_{j \in J} r_j t_j.$$

Упорядочивание задач в алгоритмах NFDH и FFDH выполнять сортировкой подсчетом (counting cort) [2, C. 223].

При реализации алгоритма FFDH рекомендуется использовать дерево турнира (tournament tree, max winner tree). Каждый лист такого дерева – уровень упаковки, а значение листа — количества свободных ЭМ на уровне. Каждый внутренний узел дерева содержит максимальное из значений левого и правого дочерних узлов. В таком дереве поиск *первого подходящего уровня* (first fit, FF) выполняется за время $O(\log m)$. Последнее обеспечивает выполнение алгоритма FFDH за время $O(m \log m)$.

Источники информации об алгоритмах FF, NFDH и FFDH:

- David S. Johnson. Case Studies: Bin Packing & The Traveling Salesman Problem

  `http://logic.pdmi.ras.ru/midas/sites/default/files/Johnson-Tuesday.pdf`
- Heidi Smith. Level Algorithms and Shelf Algorithms for //

  `http://users.cs.cf.ac.uk/C.L.Mumford/heidi/Approaches.html`
- Jan H van Vuuren. 2D Strip Packing Problem //

  `http://dip.sun.ac.za/~vuuren/repositories/levelpaper/spp[1].htm`

2. **Исследовать** время выполнения алгоритмов в зависимости от количества m задач в наборе.

Сформировать 10 наборов задач с $m = 500, 1000, \ldots, 5000$; параметры задач генерировать как равномерно распределенные псевдослучайные числа $r_j \in \{1, 2, \ldots, n\}$, $t_j \in \{1, 2, \ldots, 100\}$. Рассмотреть случаи для $n = 1024$ и $n = 4096$.

3. **Провести сравнительный анализ** значений целевой функции от расписаний, формируемых алгоритмами.

Сформировать 10 наборов задач ($m = 500, 1000, \ldots, 5000$); параметры задач генерировать как равномерно распределенные псевдослучайные числа $r_j \in \{1, 2, \ldots, n\}$, $t_j \in \{1, 2, \ldots, 100\}$. Во всех 10 экспериментах $n = 1024$. По результатам экспериментов построить оценки математического ожидания и среднеквадратического отклонения величины ε (см. приложение).

4. **Выполнить сравнительный анализ** значений целевой функции от расписаний, формируемых алгоритмами.

На основе протоколов решения параллельных задач на промышленных распределенных ВС сформировать наборы задач ($m = 500, 1000, 1500$) для любой из систем: LLNL uBGL, LLNL Atlas, LLNL Thunder. По результатам экспериментов построить оценки математического ожидания и среднеквадратического отклонения величины ε.

Входные данные (статистика выполнения параллельных программ на реальных системах):

- Logs of Real Parallel Workloads from Production Systems // `http://www.cs.huji.ac.il/labs/parallel/workload/logs.html`.

3.1.3. Контрольные вопросы

1. Дать определение мультипрограммного режима функционирования распределенных ВС. Назвать цели организации функционирования ВС.

2. Объяснить, в чем заключается задача построения расписания решения параллельных задач на распределенной ВС.

3. Пояснить содержательный смысл целевой функции (3.1).

4. Пояснить смысл ограничений в оптимизационной задаче (3.1)–(3.4).

5. Объяснить, почему задача (3.1)–(3.4) относится к трудноразрешимым задачам дискретной оптимизации.

6. Какова вычислительная сложность реализованных алгоритмов NFDH и FFDH?

7. Какой из алгоритмов на рассмотренных наборах задач формировал более точные расписания?

8. Проанализировать, от чего зависит эффективность того или иного алгоритма упаковки. Как значения параметров m и n влияют на время работы алгоритмов?

3.2. Теоретико-игровой подход к обслуживанию потока задач

3.2.1. Игра «диспетчер-вычислительный центр»

Имеется вычислительный центр (ВЦ), эксплуатирующий распределенную вычислительную систему из n элементарных машин. ВЦ для решения задач может выставлять по своему усмотрению любое число $i \in E$, $E = \{0, 1, \ldots, n\}$ ЭМ.

Считается, что задача имеет ранг $j \in E$, если для ее выполнения требуется j машин. Предполагается, что в очереди диспетчера присутствуют задачи всех рангов. Время решения задачи i равно t_i единиц времени. Будем считать, что диспетчер в дискретные моменты времени $t = 0, 1, 2, \ldots$ назначает на ВС задачи различных рангов с временем решения, равным 1.

Рассмотрим игру с участием двух игроков: ВЦ и диспетчера. Будем говорить, что ВЦ использует чистую стратегию с номером $i \in E$, если он для решения задач отводит i машин, и что диспетчер использует чистую стратегию $j \in E$, если он для решения на ВС назначает задачу с рангом j. Если ВЦ выбирает стратегию с номером i, а диспетчер – стратегию с номером j, то диспетчер «платит» ВЦ сумму c_{ij}. Элементы c_{ij}, $i, j \in E$ составляют матрицу платежей $\mathbf{C}$.

Если ВЦ применяет смешанную стратегию $P = (p_0, p_1, \ldots, p_n)$, а диспетчер – смешанную стратегию $\Pi = (\pi_0, \pi_1, \ldots, \pi_n)$, то средний платеж вычислительному центру составляет

$$\sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} p_i \pi_j.$$

Здесь p_i, π_j – вероятности выбора соответственно вычислительным центром стратегии с номером i и диспетчером стратегии с номером j.

ВЦ имеет оптимальную смешанную стратегию $P^* = (p_0^*, p_1^*, \ldots, p_n^*)$, а диспетчер – оптимальную смешанную стратегию $\Pi = (\pi_0^*, \pi_1^*, \ldots, \pi_n^*)$ такие, что

$$\sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} p_i \pi_j^* \le V \le \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} p_i^* \pi_j,$$

$$V = \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} p_i^* \pi_j^*.$$

Требуется для заданной матрицы платежей найти решение – векторы P^*, Π^* и цену V игры.

Подбор элементов платежной матрицы $\mathbf{C}$ должен осуществляться с учетом конкретных условий эксплуатации ВС. Будем считать, что если i, j – чистые стратегии соответственно ВЦ и диспетчера, то элементы матрицы платежей

$$c_{ij} = \begin{cases} jc_1 + (i-j)c_2, & \text{при } i \geq j, \\ ic_2 + (j-i)c_3, & \text{при } i < j, \end{cases}$$

где c_1 – платеж за использование одной машины в течение единицы времени, c_2 и c_3 – штрафы в единицу времени соответственно за простой одной машины и при $j - i = 1$, $i, j \in E$.

Имеет место теорема [3, С. 187].

Теорема. *Матрица* c_{ij} *не имеет седловых точек тогда и только тогда, когда* $c_1 < \min\{c_2, c_3\}$.

3.2.2. Итеративный метод Брауна

Одним из методов решения матричных игр является итеративный метод Брауна [3, С. 192].

Обозначим через $\mathbf{C} = ||c_{ij}||$ $n \times n$-матрицу платежей; C_i – i-я строка матрицы $\mathbf{C}$, а C_j – ее j-й столбец.

Рассмотрим последовательности векторов:

$$X(0), X(1), \ldots, X(l), \ldots,$$

$$Y(0), Y(1), \ldots, Y(l), \ldots,$$

где

$$X(l) = (x_0(l), x_1(l), \ldots, x_n(l)),$$
$$Y(l) = (y_0(l), y_1(l), \ldots, y_n(l)).$$

Компонента $x_i(l)$, $i \in E$ – это относительная оценка выигрыша ВЦ, если он в l-й итерации выбирает i-ю строку матрицы $\mathbf{C}$; $y_j(l)$ – относительная оценка выигрыша диспетчера, если он в l-й итерации выбирает j-й столбец матрицы $\mathbf{C}$.

На итерации l ВЦ и диспетчер выбирают соответственно строку i и столбец j такие, что

$$x_i(l) = \max_{0 \leq i \leq n} x_i(l) = \max X(l),$$

$$y_i(l) = \max_{0 \leq i \leq n} y_i(l) = \max Y(l).$$

Учитывая таким образом найденные i и j, игроки пересматривают свои оценки значений строк и столбцов для следующей $(l+1)$-й итерации:

$$X(l+1) = X(l) + C_{\cdot j},$$

$$Y(l+1) = Y(l) + C_{i\cdot}.$$

Нетрудно показать, что

$$X(l) = X(0) + \sum_{j=0}^{n} l\pi_j(l) C_{\cdot j},$$

$$Y(l) = Y(0) + \sum_{i=0}^{n} lp_i(l) C_{i\cdot},$$

где $l\pi_j$ равно числу выборов столбца j, а $lp_i(l)$ – числу выборов строки i в матрице $\mathbf{C}$ при реализации l-й итерации.

Векторы $P(l) = (p_0, p_1, \ldots, p_n)$ и $\Pi = (\pi_0, \pi_1, \ldots, \pi_n)$ будут оценками смешанных стратегий ВЦ и диспетчера, которые сходятся к оптимальным стратегиям P^* и Π^*. Итак, при больших l

$$P^* \approx P(l), \quad \Pi^* \approx \Pi(l),$$

$$V \approx \frac{\max X(l) - \max Y(l)}{2l} \leq V(l),$$

где $V(l)$ – это значение цены игры после l-й итерации.

Практически можно положить

$$X(0) = Y(0) = 0.$$

В качестве меры близости $V(l)$ к V можно взять

$$\frac{\max X(l) - \max Y(l)}{l} \leq \varepsilon,$$

где $\varepsilon > 0$.

3.2.3. Задание

1. **Разработать программу** решения методом Брауна игровой задачи «диспетчер-вычислительный центр».

Входные данные n, c_1, c_2, c_3 загружаются из файла или указываются как аргументы в командной строке. Для формирования матрицы платежей использовать подход, описанный в [3, С. 187]. Значения c_1, c_2, c_3 выбирать следующим образом:

$$c_1 \in \{1, 2, 3\}, \quad c_2, c_3 \in \{4, 5, 6\} \quad (\text{учесть, что } c_1 < \max\{c_2, c_3\}).$$

2. **Построить график** зависимости времени работы алгоритма от количества n элементарных машин в системе.

3. **Объяснить** функционирование ВЦ и диспетчера в соответствии с найденными оптимальными смешанными стратегиями.

3.2.4. Пример входных и выходных данных

Входные данные:

$$n = 10, \quad c_1 = 1.0, \quad c_2 = 2.0, \quad c_3 = 3.0, \quad \varepsilon = 0.01.$$

Матрица платежей **C**:

```
 0.00  3.00  6.00  9.00 12.00 15.00 18.00 21.00 24.00 27.00 30.00
 2.00  1.00  5.00  8.00 11.00 14.00 17.00 20.00 23.00 26.00 29.00
 4.00  3.00  2.00  7.00 10.00 13.00 16.00 19.00 22.00 25.00 28.00
 6.00  5.00  4.00  3.00  9.00 12.00 15.00 18.00 21.00 24.00 27.00
 8.00  7.00  6.00  5.00  4.00 11.00 14.00 17.00 20.00 23.00 26.00
10.00  9.00  8.00  7.00  6.00  5.00 13.00 16.00 19.00 22.00 25.00
12.00 11.00 10.00  9.00  8.00  7.00  6.00 15.00 18.00 21.00 24.00
14.00 13.00 12.00 11.00 10.00  9.00  8.00  7.00 17.00 20.00 23.00
16.00 15.00 14.00 13.00 12.00 11.00 10.00  9.00  8.00 19.00 22.00
18.00 17.00 16.00 15.00 14.00 13.00 12.00 11.00 10.00  9.00 21.00
20.00 19.00 18.00 17.00 16.00 15.00 14.00 13.00 12.00 11.00 10.00
```

Количество итераций $l = 1253901$.
Цена игры $V = 0.005$.
Оптимальные смешанные стратегии ВЦ:

```
0.25 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.75
```

Оптимальные смешанные стратегии диспетчера:

```
0.31 0.00 0.00 0.03 0.03 0.00 0.24 0.01 0.18 0.14 0.06
```

3.2.5. Контрольные вопросы

1. Дать определение режиму обслуживания потока задач.

2. Объяснить, в чем заключается положительные стороны теоретико-игрового подхода к организации функционирования ВС (границы его применимости).

3. Дать определения следующим понятиям: решение игры, чистая стратегия, смешанная стратегия, седловая точка.

4. Объяснить, в чем заключается суть игры между вычислительным центром (ВЦ) и диспетчером. Что определяют смешанные стратегии игры?

5. Пояснить шаги итеративного метода Брауна для решения прямоугольных игр.

3.3. Стохастически оптимальное функционирование ВС

3.3.1. Задача формирования подсистем ЭМ

Имеется распределенная вычислительная система, состоящая из n элементарных машин. Программным путем система может быть разбита на n подсистем различных рангов: из одной машины, из двух, ..., из n машин. В таких логически изолированных подсистемах могут одновременно выполняться параллельные программы.

На вход в систему поступает поток параллельных задач различных рангов. Пусть спрос a_j на подсистему ранга j есть непрерывная случайная величина с плотностью распределения вероятностей $p_j(a_j)$. Тогда математическим ожиданием спроса на подсистему ранга j будет

$$\rho_j = \int\limits_0^\infty a_j p_j(a_j) da_j.$$

Обозначим через d_j цену эксплуатации, а за c_j – стоимость эксплуатации подсистемы ранга j в течение длительного промежутка времени T.

Если спрос на подсистему ранга j за время T превысит число организованных подсистем ранга j, то убыток составит $d_j - c_j$ за каждый неудовлетворенный спрос. С другой стороны, если организовано подсистем ранга j больше, чем требуется, то убыток составит c_j на каждую избыточную подсистему.

Требуется найти значения неизвестных $(x_1, x_2, \ldots, x_n)$, где x_j – количество организуемых подсистем ранга j. Разбиение $(x_1, x_2, \ldots, x_n)$ должно максимизировать ожидаемую прибыль за время T [3, С. 196].

Ожидаемые потери от недостатка подсистем ранга j составят

$$(d_j - c_j) \int\limits_{x_j}^\infty (a_j - x_j) p_j(a_j) da_j,$$

а ожидаемые потери от избытка

$$c_j \int\limits_0^{x_j} (x_j - a_j) p_j(a_j) da_j = c_j(x_j - \rho_j) + c_j \int\limits_{x_j}^\infty (a_j - x_j) p_j(a_j) da_j.$$

Математическое ожидание прибыли при эксплуатации ВС равно

$$\sum_{j=1}^{n}(d_j - c_j)\rho_j - \sum_{j=1}^{n} c_j(x_j - \rho_j) - \sum_{j=1}^{n} d_j \int\limits_{x_j}^{\infty}(a_j - x_j)p_j(a_j)da_j,$$

или

$$\sum_{j=1}^{n}(d_j\rho_j - c_jx_j - d_j \int\limits_{x_j}^{\infty}(a_j - x_j)p_j(a_j)da_j).$$

Итак, требуется найти разбиение $(x_1, x_2, \ldots, x_n)$ системы на подсистемы, доставляющее минимум целевой функции f

$$f(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n}(c_jx_j - d_j \int\limits_{x_j}^{\infty}(a_j - x_j)p_j(a_j)da_j) \to \min_{(x_j)} \tag{3.5}$$

при ограничениях:

$$\sum_{j=1}^{n} jx_j \le n, \tag{3.6}$$

$$x_j \ge 0, \quad j = 1, 2, \ldots, n. \tag{3.7}$$

Сформулированная задача может быть решения методом динамического программирования [3, С. 207].

3.3.2. Метод динамического программирования

Ниже приведена оптимизационная задача (3.8)–(3.10) с аддитивной целевой функцией $f(x_1, x_2, \ldots, x_n)$. В [3, 4] показано, что такая задача может быть точно решена методом динамического программирования.

$$f(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} f_j(x_j) \to \min_{(x_j)} \tag{3.8}$$

при ограничениях:

$$\sum_{j=1}^{n} a_jx_j \le b, \tag{3.9}$$

$$x_j \ge 0, \quad j = 1, 2, \ldots, n. \tag{3.10}$$

Нетрудно заметить, что задача (3.5)–(3.7) может быть приведена к виду задачи (3.8)–(3.10). Для этого достаточно положить

$$f_j(x_j) = c_j x_j - d_j \int\limits_{x_j}^{\infty} (a_j - x_j) p_j(a_j) da_j,$$

$$a_j = j, \quad j = 1, 2, \ldots, n,$$
$$b = n.$$

Рассмотрим основные шаги метода динамического программирования для решения задачи (3.8)–(3.10) [3, С. 207].

Можно показать, что для последовательности функций

$$\Lambda_k(\xi) = \min_{x_1,\ldots,x_k} \sum_{j=1}^{k} f_j(x_j), \quad k = 1, 2, \ldots, n, \quad \xi = 0, 1, \ldots, b,$$

в которых минимум берется по неотрицательным целым числам, удовлетворяющим условию

$$\sum_{j=1}^{n} a_j x_j \leq \xi,$$

справедливы рекуррентные соотношения:

$$\Lambda_k(\xi) = \min_{x_k} (f_k(x_k) + \Lambda_{k-1}(\xi - a_k x_k)), \tag{3.11}$$

где x_k может принимать значения $0, 1, \ldots, [\xi/a_k]$. Запись $[x]$ – это целая часть числа x (см. приложение).

На *прямом проходе* отыскивается оптимальное значение z^* целевой функции

$$z^* = f(x_1^*, x_2^*, \ldots, x_n^*)$$

Вычислительная процедура для отыскания z^* состоит в непосредственном определении $\Lambda_1(\xi)$, затем в вычислении $\Lambda_k(\xi)$, $k = 2, 3, \ldots, n-1$, и $z^* = \Lambda_n(b)$. Результаты заносятся в таблицу 3.1.

Сперва находятся $\Lambda_1(\xi)$ и $\widehat{x}_1(\xi)$ для всех $\xi = 0, 1, \ldots, b$.

$$\Lambda_1(\xi) = \min_{x_1} f_1(x_1), \quad x_1 = 0, 1, \ldots, [\xi/a_1].$$

Величина $\widehat{x}_1(\xi)$ – значение $x_1 \in \{0, 1, \ldots, [\xi/a_1]\}$, при котором $\Lambda_1(\xi)$ принимает минимальное значение.

$$\widehat{x}_1(\xi) = \operatorname*{argmin}_{x_1} f_1(x_1), \quad x_1 = 0, 1, \ldots, [\xi/a_1].$$

Найдя $\Lambda_1(\xi)$, переходим к вычислению $\Lambda_2(\xi)$ и $\widehat{x}_2(\xi)$

$$\Lambda_2(\xi) = \min_{x_2}(f_2(x_2) + \Lambda_1(\xi - a_2x_2)), \quad x_2 = 0, 1, \ldots, [\xi/a_2].$$

$$\widehat{x}_2(\xi) = \operatorname*{argmin}_{x_2}(f_2(x_2) + \Lambda_1(\xi - a_2x_2)), \quad x_2 = 0, 1, \ldots, [\xi/a_2].$$

Значение $\Lambda_1(\xi - a_2x_2)$ повторно вычислять не следует, его можно взять из таблицы 3.1.

Вычисления продолжатся вплоть до нахождения $\Lambda_n(\xi)$ и $\widehat{x}_n(\xi)$. Оптимальное значение целевой функции берется из таблицы

$$z^* = \Lambda_n(b).$$

На *обратном ходе* отыскиваются компоненты вектора $(x_1^*, x_2^*, \ldots, x_n^*)$ оптимального решения. Значение x_n^* берется из таблицы

$$x_n^* = \widehat{x}_n(b).$$

Значения остальных неизвестных вычисляются с учетом ограничений задачи и берутся из таблицы

$$x_{n-i}^* = \widehat{x}_{n-i}\left(b - \sum_{k=0}^{i-1} a_{n-k}x_{n-k}^*\right), \quad i = 1, 2, \ldots, n-1.$$

Таблица 3.1

ξ	$\Lambda_1(\xi)$	$\widehat{x}_1(\xi)$	$\Lambda_2(\xi)$	$\widehat{x}_2(\xi)$	...	$\Lambda_n(\xi)$	$\widehat{x}_n(\xi)$
0	$\Lambda_1(0)$	$\widehat{x}_1(0)$	$\Lambda_2(0)$	$\widehat{x}_2(0)$	...	$\Lambda_n(0)$	$\widehat{x}_n(0)$
1	$\Lambda_1(1)$	$\widehat{x}_1(1)$	$\Lambda_2(1)$	$\widehat{x}_2(1)$	...	$\Lambda_n(1)$	$\widehat{x}_n(1)$
...	...	...	...	...	...	...	...
b	$\Lambda_1(b)$	$\widehat{x}_1(b)$	$\Lambda_2(b)$	$\widehat{x}_2(b)$	...	$\Lambda_n(b)$	$\widehat{x}_n(b)$

3.3.3. Задание

1. **Написать программу**, реализующую метод динамического программирования для решения рассмотренной задачи.

В качестве входных параметров программа получает количество n машин в системе, значения $c_1, c_2, \ldots, c_n$, $d_1, d_2, \ldots, d_n$, $\rho_1, \rho_2, \ldots, \rho_n$. Считать, что спрос на подсистему ранга j имеет пуассоновское распределение с параметром ρ_j.

2. **Исследовать** зависимость значения целевой функции f от значений параметров c_j и d_j.

3.3.4. Контрольные вопросы

1. Объяснить суть подхода к организации функционирования распределенных ВС с использованием аппарата стохастического программирования. В чем преимущество данного подхода?

2. Привести примеры постановок задач стохастического программирования.

3. Когда следует осуществлять новое разбиение системы на подсистемы элементарных машин?

4. Объяснить основные этапы в выводе целевой функции f.

5. Пояснить основные шаги метода динамического программирования.

3.4. Вложение параллельных программ в ВС

3.4.1. Модель ВС с иерархической организацией коммуникационной среды

Пусть имеется распределенная ВС, представленная в виде дерева, содержащего L коммуникационных уровней (рис. 3.2). Рассмотрим вложение параллельной программы в подсистему из N элементарных машин. Очевидно, что подсистема также имеет иерархическую структуру.

Обозначим через n_l количество элементов на уровне $l \in \{1, 2, \ldots, L\}$ (вычислительные узлы, процессоры и пр.); n_{lk} – число прямых дочерних узлов элемента $k \in \{1, 2, \ldots, n_l\}$, находящегося на уровне l; c_{lk} – количество ЭМ, принадлежащих потомкам данного элемента.

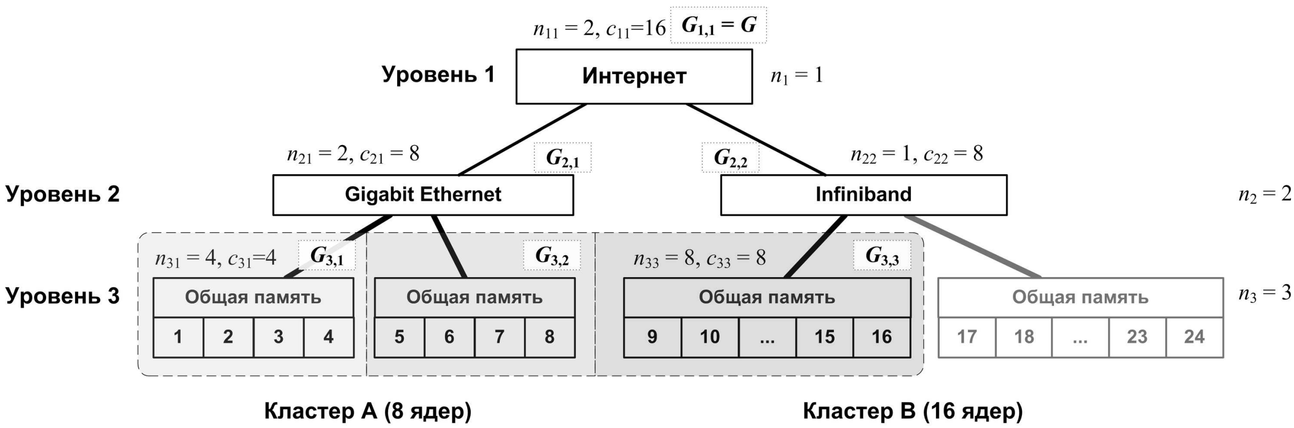

Рис. 3.2. Пример подсистемы ЭМ, выделенной для решения параллельной задачи ранга $N = 16$.

Параллельная программа представлена графом $G = (V, E)$ информационных обменов, где $V = \{1, 2, \ldots, N\}$ – множество ветвей параллельной программы, а $E \subseteq V \times V$ – множество информационно-логических связей между ветвями. Обозначим через d_{ij} вес ребра $(i, j) \in E$, отражающий интенсивность обменов данными между ветвями i и j при выполнении программы.

Вложение параллельной программы в ВС задается значениями переменных $x_{ij} \in \{0, 1\}$: $x_{ij} = 1$, если ветвь $i \in V$ назначена на ЭМ, в противном случае $x_{ij} = 0$.

Для оценки эффективности вложения используется время $T(X)$ выполнения информационных обменов. Оно определяется максимальным из времен выполнения обменов ветвями программы. Пусть $t(i, j, p, q)$ – суммарное время взаимодействий между ветвями $i, j \in V$, назначенными на процессорные ядра p и q соответственно.

Тогда
$$T(X) = \max_{i \in V} \left\{ \sum_{j=1}^{N} \sum_{p=1}^{N} \sum_{q=1}^{N} x_{ip} x_{jq} t(i,j,p,q) \right\}.$$

Значение функции $t(i,j,p,q)$ может быть получено согласно модели Хокни:
$$t(i,j,p,q) = \frac{d_{ij}}{b(p,q)},$$
где $b(p,q)$ – пропускная способность канала связи между процессорами p и q.

Сформулируем задачу оптимального вложения параллельной программы в ВС с иерархической организацией:
$$T(X) = \max_{i \in V} \left\{ \sum_{j=1}^{N} \sum_{p=1}^{N} \sum_{q=1}^{N} x_{ip} x_{jq} t(i,j,p,q) \right\} \to \min_{(x_{ij})} \qquad (3.12)$$
при ограничениях:
$$\sum_{j=1}^{N} x_{ij} = 1, \quad i = 1, 2, \ldots, N, \qquad (3.13)$$
$$\sum_{i=1}^{N} x_{ij} = 1, \quad j = 1, 2, \ldots, N, \qquad (3.14)$$
$$x_{ij} \in \{0, 1\}. \qquad (3.15)$$

Ограничения (3.13) и (3.15) гарантируют назначение каждой ветви параллельной программы на единственную ЭМ. Ограничение (3.14) обеспечивает назначение на машину одной ветви. Задача (3.12)–(3.15) относится к дискретной оптимизации и является трудноразрешимой.

3.4.2. Эвристический метод вложения параллельных программ в ВС

Метод HierarchicMap [5] вложения параллельных программ основан на рекурсивном разбиении графа параллельной программы на подмножества интенсивно обменивающихся параллельных ветвей и отображения их на ЭМ, связанные быстрыми каналами связи. Цель разбиения – минимизация суммы весов ребер, инцидентных разным подмножествам разбиения.

Разбиение выполняется многократно для каждого уровня иерархии коммуникационной среды. Функция PartGraph возвращает список подграфов, получаемых в результате разбиения исходного графа.

Ниже приведен псевдокод метода HierarchicMap, на вход которого подается граф G, номер l уровня коммуникационной среды и номер k элемента на уровне l.

Проиллюстрируем метод на примере отображения параллельной программы на подсистему из 16 ЭМ (рис. 3.2). На первом шаге выполняется разбиение (PartGraph) исходного графа G на n_{11} подграфов (G_{21} и G_{22}) по c_{21} и c_{22} вершин. Далее графы G_{21} и G_{22} рекурсивно разбиваются на n_{21} и n_{22} частей. Полученные в результате этого разбиения подграфы G_{31}, G_{32}, G_{33} (их вершины – ветви программы) назначаются на узел 1 (процессорные ядра $1, 2, 3, 4$) и узел 2 (процессорные ядра 5, 6, 7, 8) кластера A и узел 1 (процессорные ядра $9, 10, \ldots, 16$) кластера B.

```
function HierarchicMap(G, l, k)
    if l = L then
        return G_{L,1}, G_{L,2}, ..., G_{L,n_L}
    else
        (G_{l+1,1}, ..., G_{l+1,n_{lk}}) = PartGraph(G_{lk}, n_{lk}, c_{l+1,1}, ..., c_{l+1,n_{lk}})
        for k = 1 to n_{lk} do
            HierarchicMap(G_{l+1,k}, l + 1, k)
        end for
    end if
end function
```

3.4.3. Задание

1. **Написать программу**, реализующую метод HierarchicMap вложения параллельных программ в распределенные ВС.

2. **Построить графики** зависимости времени выполнения алгоритма и значения целевой функции $T(X)$ от числа N параллельных ветвей для разных алгоритмов разбиения графов.

Для разбиения графов рекомендуется использовать пакеты METIS или Scotch. Информационные графы параллельных программ для различного числа N параллельных ветвей находятся на сетевом ресурсе, доступ к которому следует получить у преподавателя.

В качестве модельной ВС использовать конфигурацию вычислительного кластера на базе SMP-узлов. В котором: уровень 1 – сеть связи между серверными стойками (InfiniBand QDR), уровень 2 – сеть связи внутри стоек между узлами (InfiniBand QDR), уровень 3 – разделяемая память узлов (DDR3-1600). Каждый вычислительный узел содержит два четырехъядерных процессора. Подсистемы формируются из полностью выделенных для решения задачи вычислительных узлов.

В качестве значений пропускной способности $b(p, q)$ между процессорными ядрами рекомендуется взять пиковую пропускную способность (оценку сверху) для соответствующего канала связи.

3.4.4. Контрольные вопросы

1. Описать модель распределенной ВС с иерархической структурой.

2. Что такое информационный граф задачи? Как его получить для параллельных программ в стандарте MPI?

3. Дать определение задачи вложения. Почему важно учитывать иерархическую структуру распределенных ВС при реализации параллельных программ на них?

4. От чего зависит эффективность метода HierarchicMap? В каких случаях он неэффективен?

4. Приложения

4.1. Функции округления

Округление вещественного числа x до меньшего ближайшего целого числа обозначается как $\lfloor x \rfloor$ или $\mathrm{floor}(x)$ – *пол*. Также эту функцию называют *антье* (от фр. entier) – целая часть вещественного числа

$$x - 1 < \lfloor x \rfloor \leq x.$$

В литературе можно встретить альтернативное обозначение целой части части числа x (функции пол) – это обозначение Гаусса $[x]$.

Примеры использования функции пол:

- $\lfloor 1.3 \rfloor = 1$;
- $\lfloor 1.7 \rfloor = 1$;
- $\lfloor -3.7 \rfloor = -4$;
- $\lfloor -4 \rfloor = -4$.

Округление вещественного числа x до большего ближайшего целого числа обозначается как $\lceil x \rceil$ или $\mathrm{ceil}(x)$ – *потолок*

$$x \leq \lceil x \rceil < x + 1.$$

Примеры использования функции потолок:

- $\lceil 1.3 \rceil = 2$;
- $\lceil 1.7 \rceil = 2$;
- $\lceil -3.7 \rceil = -3$;
- $\lceil -4 \rceil = -4$.

Функции *пол* и *потолок* введены 1962 г. К. Айверсоном [6]. Для них справедливы следующие соотношения

$$\lfloor x \rfloor \leq x \leq \lceil x \rceil.$$

Целочисленное слагаемое a можно вносить и выносить за скобки функций пол/потолок

$$\lfloor x + a \rfloor = \lfloor x \rfloor + a, \quad \lceil x + a \rceil = \lceil x \rceil + a.$$

4.2. Факториалы

Факториал (factorial) целого числа n – это произведение натуральных чисел от 1 до n включительно

$$n! = 1 \cdot 2 \cdot \ldots \cdot n,$$

$$n! = n \cdot (n-1)!,$$

$$0! = 1.$$

Асимптотическая *формула Стирлинга* для приближенного вычисления факториала целого числа n

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + O\left(\frac{1}{n}\right)\right),$$

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

4.3. Элементарная обработка результатов измерений

Пусть требуется измерить некоторую величину x. Например, время выполнения программы, реализующей некоторый алгоритм. Рассмотрим основные этапы измерения величины x и обработки полученных результатов.

1. **Выполняем серию измерений**. Для уменьшения влияния случайных ошибок выполняем n измерений величины x. Обозначим результаты измерений через

$$x_1, x_2, \ldots, x_n.$$

Такой ряд значений величины x называют *выборкой* (sample).

2. **Вычисляем характеристики выборки**. При конечном числе n измерений в качестве оценки истинного значения измеряемой величины используют среднее арифметическое $\bar{x}$ результатов измерений или, как его еще называют, *выборочное среднее* (sample mean). Вычисляем его по следующей формуле

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

Перед вычислением среднего значения $\bar{x}$ можно провести анализ выборки и отбросить *промахи измерений* (выбросы, outliers). Например, можно упорядочить выборку по значениям x_i и отбросить 25% первых и последних элементов (наименьших и наибольших).

Вычисляем *несмещенную оценку s^2 дисперсии* величины x (unbiased sample variance)

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n(n-1)}\left(n\sum_{i=1}^{n}x_i^2 - (\sum_{i=1}^{n}x_i)^2\right). \tag{4.1}$$

Отклонение s отдельного результата измерения x_i от среднего значения $\bar{x}$ называют *средней квадратичной ошибкой* или *средней квадратической ошибкой* (sample standard deviation, StdDev)

$$s = \sqrt{s^2} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} = \sqrt{\frac{1}{n(n-1)}\left(n\sum_{i=1}^{n}x_i^2 - (\sum_{i=1}^{n}x_i)^2\right)}.$$

Среднеквадратичной ошибкой среднего арифметического (standard error of the mean, StdErr) называется величина $s_{\bar{x}}$, которая характеризует точность, с которой получено среднее значение $\bar{x}$. Вычисляем $s_{\bar{x}}$ по следующей формуле

$$s_{\bar{x}} = \frac{s}{\sqrt{n}} = \sqrt{\frac{s^2}{n}} = \sqrt{\frac{\sum\limits_{i=1}^{n}(x_i - \bar{x})^2}{n(n-1)}}.$$

3. **Строим доверительный интервал**. Для расчета абсолютной ошибки при малом количестве измерений вводится специальный коэффициент, зависящий от надежности (вероятности) p и числа измерений n, называемый *коэффициентом Стьюдента* $t_{p,n}$.

Задаем уровень доверительной вероятности (например, $p = 0.99$) и по табл. 4.1 находим значение коэффициента $t_{p,n}$. Вычисляем *абсолютную ошибку* Δx

$$\Delta x = s_{\bar{x}} \cdot t_{p,n}.$$

Доверительный интервал (confidence interval) записываем в виде

$$x = \bar{x} \pm s_{\bar{x}} \cdot t_{p,n}.$$

Последняя запись говорит о том, что с вероятностью p истинное значение измеренной величины x лежит в интервале

$$[\bar{x} - s_{\bar{x}} \cdot t_{p,n}, \bar{x} + s_{\bar{x}} \cdot t_{p,n}].$$

Таблица 4.1. Значения коэффициентов Стьюдента $t_{p,n}$

n	p		
	0.95	0.99	0.999
2	12.706	63.657	636.61
3	4.303	9.925	31.598
4	3.182	5.841	12.941
5	2.776	4.604	8.610
6	2.571	4.032	6.859
7	2.447	3.707	5.959
8	2.365	3.499	5.405
9	2.306	3.355	5.041
10	2.262	3.250	4.781
20	2.093	2.861	3.883
30	2.045	2.756	3.659
40	2.021	2.704	3.551
50	2.021	2.704	3.551
∞	1.960	2.576	3.291

4.3.1. Метод Велфорда вычисления среднего квадратичного отклонения

При вычислении s^2 по формуле (4.1) значение s^2 может быть меньше нуля. Следовательно, при вычислении $s = \sqrt{s^2}$ мы получим ошибку. В работе [7, С. 259] для нахождения значения s рассмотрен метод Велфорда (B.P. Welford), основанный на следующих рекуррентных соотношениях

$$M_1 = x_1, \quad M_k = M_{k-1} + (x_k - M_{k-1})/k,$$

$$S_1 = 0, \quad S_k = S_{k-1} + (x_k - M_{k-1}) \cdot (x_k - M_K),$$

для $k = 2, 3, \ldots, n$.

Откуда

$$s = \sqrt{S_n/(n-1)}.$$

4.4. Вопросы к экзамену

4.4.1. Надежность и живучесть вычислительных систем

1. Архитектурные особенности современных распределенных вычислительных систем. Режимы функционирования вычислительных систем.

2. Модели вычислительной системы со структурной избыточностью. Показатели надежности вычислительных систем в переходном и стационарном режимах функционирования.

3. Методика расчета показателей надежности вычислительных систем в переходном и стационарном режимах функционирования.

4. Живучие вычислительные системы. Потенциальная и структурная живучесть. Показатели потенциальной живучести вычислительных систем.

5. Расчет показателей живучести вычислительных систем. Континуальный подход к анализу живучести вычислительных систем.

6. Показатели структурной живучести распределенных вычислительных систем. Перспективные структуры распределенных вычислительных систем.

7. Осуществимость решения задач на вычислительных системах.

8. Технико-экономическая эффективность функционирования вычислительных систем.

9. Экспресс-анализ функционирования вычислительных систем.

4.4.2. Организация функционирования вычислительных систем

1. Цели и задачи организации функционирования распределенных вычислительных систем.

2. Мультипрограммные режимы функционирования распределенных вычислительных систем. Режим обработки набора параллельных задач. Режим обслуживания потока параллельных задач.

3. Методы построения расписаний выполнения параллельных программ на ВС.

4. Эвристические алгоритмы упаковки прямоугольников в полуограниченную полосу. Вычислительная сложность алгоритмов.

5. Теоретико-игровой подход к организации функционирования распределенных вычислительных систем. Игра «диспетчер-вычислительный центр».

6. Методы решения теоретико-игровых задач. Итеративный метод Брауна.

7. Подход к организации функционирования распределенных вычислительных систем с привлечением аппарата стохастического программирова-

ния.

8. Методы решения задач стохастического программирования при обслуживании потоков параллельных программ.

9. Децентрализованные алгоритмы обслуживания потоков задач в распределенных вычислительных системах.

10. Вложение в распределенные вычислительные системы параллельных программ с целью минимизации времени их выполнения.

11. Задача разбиения вычислительной системы на подсистемы элементарных машин.

12. Алгоритмы реализации коллективных информационных обменов в распределенных вычислительных системах.

Литература

[1] Хорошевский В. Г. Архитектура вычислительных систем – 2-е изд. — М. : МГТУ им. Н.Э. Баумана, 2008. — 520 с.

[2] Алгоритмы: построение и анализ. – 3-е изд. / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн. — М. : Вильямс, 2013. — 1328 с.

[3] Евреинов Э. В., Хорошевский В. Г. Однородные вычислительные системы. — Н. : Наука. Сибирское отд-е, 1978. — 319 с.

[4] Хедли Дж. Нелинейное и динамическое программирование. — М. : Мир, 1967. — 508 с.

[5] Курносов М. Г., Пазников А. А. Эвристические алгоритмы отображения параллельных MPI-программ на мультикластерные вычислительные и GRID-системы // Вычислительные методы и программирование. — 2013. — № 14. — С. 1–10.

[6] Грэхем Р., Кнут Д.Э., Паташник О. Конкретная математика. Математические основы информатики. — М. : Вильямс, 2010. — 784 с.

[7] Кнут Д.Э. Искусство программирования. Том 2. Получисленные алгоритмы. — М. : Вильямс, 2011. — 832 с.

[8] Хорошевский В. Г. Инженерный анализ функционирования вычислительных машин и систем. — М. : Радио и связь. — 256 с.

[9] Кнут Д.Э. Искусство программирования. Том 1. Основные алгоритмы. — М. : Вильямс, 2010. — 720 с.

[10] Кнут Д.Э. Искусство программирования. Том 3. Сортировка и поиск. — М. : Вильямс, 2012. — 824 с.

[11] М. Гэри, Д. Джонсон. Вычислительные машины и труднорешаемые задачи. — М. : Мир, 1982. — 416 с.

[12] С. Танаев В., В. Шкурба В. Введение в теорию расписаний. — М. : Наука, 1975. — 256 с.

Практикум

Курносов Михаил Георгиевич
Пазников Алексей Александрович

Основы теории функционирования распределенных вычислительных систем

Подписано в печать 25.09.2015. Формат $70 \times 100\,^1/_{16}$.
Бумага офсетная. Печать цифровая.
Усл. печ. л. 4,19. Уч.-изд. л. 1,6.
Тираж 300. Заказ 0925.

Отпечатано в ООО «Автограф».
630090, Новосибирск, Весенний пр., 4. Тел. (383) 330-26-98.