УДК 004.942

ДОСЛІДЖЕННЯ АЛГОРИТМІВ ДИСПЕТЧЕРИЗАЦІЇ В КОМП'ЮТЕРНИХ СИСТЕМАХ

В.Б. Кропивницька, Б.В. Клим, А.Г. Романчук, М.О. Слабінога

ІФНТУНГ, 76019, м. Івано-Франківськ, вул. Карпатська, 15, тел. (03422) 46067,
e-mail: p u b l i c @ n u n g . e d u . u a

Розроблено систему імітаційного моделювання поведінки процесів в комп'ютерних системах (зокрема, нафтогазового комплексу) з урахуванням заданих алгоритмів диспетчеризації, що дозволяє проводити моделювання обробки даних процесів системою і тим самим визначити оптимальні характеристики та стратегії диспетчера системи для роботи з заданим набором процесів.

Зроблено аналіз існуючих алгоритмів планування процесів, виділені їхні недоліки та переваги. Визначено роль диспетчеризації в комп'ютерних системах, які тут розглядають як системи масового обслуговування. Розглянуті основні алгоритми планування і проведено дослідження їхнього впливу на завершення процесів в системі за допомогою розробленої програми імітаційного моделювання.

Ключові слова: алгоритми диспетчеризації, процес, комп'ютерна система, імітаційне моделювання.

Разработана система имитационного моделирования поведения процессов в компьютерных системах (в частности нефтегазового комплекса) с учетом заданных алгоритмов диспетчеризации, что позволяет проводить моделирование обработки данных процессов системой и тем самым определить оптимальные характеристики и стратегии диспетчера системы для работы с заданным набором процессов.

Проанализированы существующие алгоритмы планирования процессов, выделены их достоинства и недостатки. Определена роль диспетчеризации в компьютерных системах, которые рассматриваются здесь как системы массового обслуживания. Рассмотрены основные алгоритмы планирования и проведены исследования их влияния на завершение процессов в системе при помощи созданной программы имитационного моделирования.

Ключевые слова: алгоритмы диспетчеризации, процесс, компьютерная система, имитационное моделирование.

Article deals with development of the simulation system of processes behavior in computer systems depending on given scheduling algorithms. This allows to simulate processes execution and thereby to determine the system scheduler's optimal features and strategies for a given set of processes.

In the process existing queuing models that fully characterize the processing system were analysed. The scheduling role in the systems of this type was determined. Basic scheduling algorithms were considered and their ompact on completion processes in the system was investigated wath a heep of the simulation program.

Keywords: algorithms of dispatcher, process, computer system, simulation modeling.

І Вступ. Комп'ютерні системи постійно розвиваються та стають більш потужними. В той же час розширюється набір задач, які ними вирішуються, зростає їх кількість та потреба в системних ресурсах. Це, зокрема, стосується об'єктів нафтогазового комплексу, які, як правило, розподілені в просторі, що спричиняє збільшення трафіка в комп'ютерних мережах.

Однією з ключових концепцій багатозадачності, функціонування систем загального призначення та систем реального часу є диспетчеризація. Вона призначена для розподілу ресурсів системи між багатьма задачами з потенційно конкуруючими вимогами. Кожна комп'ютерна система, чи то система пакетної обробки, інтерактивна система чи система реального часу, розподіляє наявні ресурси між процесами так, щоб оптимізувати свою роботу відповідно до одного або декількох користувацьких чи системних критеріїв. Власне, розподілом системних ресурсів займається операційна система, а способами цього розподілу – її частина, відома як планувальник або диспетчер, який при цьому повинен задовольнити певні вимоги системи, наприклад такі, як час відгуку певного процесу, пропускну здатність чи коефіцієнт використання процесора.

Імітаційне моделювання є потужним інструментом, який дає змогу визначити оптимальні стратегії диспетчера для певного набору процесів. Таке моделювання дає можливість отримати більш детальне уявлення про поведінку процесів у системі, ніж прогнозні розрахунки та розробка аналітичної моделі.

Попри значний інтерес до проблеми обслуговування процесів у комп'ютерних системах, особливо при розробці та реалізації операційних систем, проблемі імітаційного моделювання поведінки процесів у системі великої уваги не приділялось. Можна виділити лише декілька програмних продуктів, які використовуються для цієї мети: MOSS Scheduling Simulator, CPUSS, Cheddar. Кожна із перелічених систем моделювання має свої недоліки: незначна кількість алгоритмів диспетчеризації, роботу яких можна зімітувати; неможливість налаштування користувачем параметрів цих алгоритмів; мала кількість статистичних показників, які враховуються при визначенні характеристик системи; недорозвинений і незручний графічний інтерфейс або взагалі його відсутність, що вимагає часу на освоєння програми. Крім того, жоден із програмних продуктів не дозволяє прове-

сти імітаційне моделювання поведінки процесів у багатопроцесорній системі.

II Постановка завдання. Сформулюємо завдання дослідити роботу процесів у операційних системах при використанні різних алгоритмів планування. Для дослідження процесів диспетчеризації в комп'ютерних системах потрібно розробити систему імітаційного моделювання поведінки процесів, яка відтворює роботу частини операційної системи, відомої як диспетчер, що дозволило б проводити моделювання обробки процесів системою в залежності від різної кількості процесорів та з врахуванням заданих алгоритмів.

III Результати. Основна мета диспетчеризації – це розподіл процесорного часу таким чином, щоб оптимізувати один або декілька аспектів поведінки системи. Загалом, існує досить багато критеріїв оцінки різних стратегій планування.

Найбільш популярні критерії можуть бути класифіковані в двох площинах. По-перше їх можна поділити на користувацькі та системні. Користувацькі критерії пов'язані з поведінкою системи стосовно окремого користувача або процесу. Системні критерії орієнтовані на ефективність і повноту використання процесора, тоді як користувацькі критерії важливі майже для всіх систем, системні критерії, скажімо, для однокористувацьких систем не такі значущі.

Ще один спосіб розподілу критеріїв: ті, які пов'язані з продуктивністю системи, і ті, які з продуктивністю безпосередньо не пов'язані. Критерії, орієнтовані на продуктивність, виражаються числовими значеннями і зазвичай їх досить легко виміряти. Критерії ж, не пов'язані з продуктивністю або якісні за своєю природою, важко піддаються вимірюванню і аналізу [1, 2].

Найпростіший алгоритм диспетчеризації «першим прийшов – першим обслужений» – First-Come, First-Served (FCFS). Як тільки процес стає готовим до виконання, він приєднується до черги готових процесів. При зупинці виконання поточного процесу для виконання вибирається той процес, який знаходився в черзі найдовше. Такий алгоритм здійснює невитісняюче планування.

Алгоритм FCFS є дуже простим у реалізації, але він має ряд недоліків. Середній час очікування і середній повний час виконання процесу для цього алгоритму істотно залежать від порядку розташування процесів у черзі. Якщо виконується деякий тривалий процес, то короткотривалі процеси, що перейшли в стан готовності після виконання тривалого процесу, будуть дуже довго чекати на початок виконання [3]. Покажемо це на прикладі. Нехай до системи на виконання прибувають 3 процеси (табл. 1).

Залежність середнього часу очікування від порядку виконання заданих процесів показано на рис. 1.

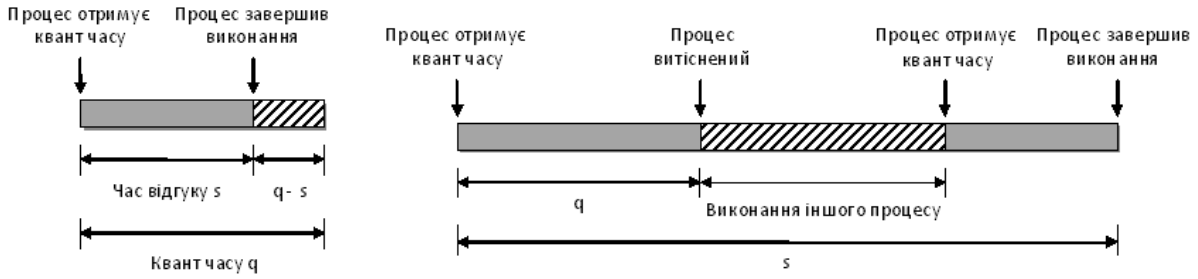
Таблиця 1 – Параметри процесів, що виконуються з використанням стратегії FCFS

Процес	Час прибуття	Час виконання
P_1	0	18
P_2	0	3
P_3	0	3

Рисунок 1 – Залежність середнього часу очікування від порядку виконання процесів для алгоритму FCFS

Інша складність при використанні стратегії FCFS пов'язана з тенденцією процесів, орієнтованих на роботу з процесором, до отримання переваги над процесами, орієнтованими на ввід-вивід. При роботі процесу, орієнтованого на процесор, решта процесів, що орієнтовані на ввід-вивід, вимушені знаходитися в стані очікування. Деякі з них можуть знаходитися в черзі вводу-виводу в заблокованому стані, або можуть повернутися в чергу готових до виконання процесів за той час, поки виконується процес, орієнтований на використання процесора. Виникає ситуація, коли попри потенційну можливість роботи пристроїв вводу-виводу вони знаходяться в стані простою. При перериванні виконання поточного процесу готові процеси (орієнтовані на ввід-вивід) швидко проходять через стан виконання і тут же виявляються заблокованими черговою операцією вводу-виводу. Якщо ж у цей момент опиниться заблокованим той процес, що орієнтований на використання процесора, то процесор знаходитиметься в стані простою і, таким чином, стратегія FCFS може спричинити неефективне використання як пристроїв вводу-виводу, так і процесора. [2]

Очевидним шляхом підвищення ефективності роботи з короткотривалими процесами в схемі FCFS є використання витіснення на основі таймера. Проста стратегія, заснована на цій ідеї, – це стратегія кругового (карусельного) планування (Round Robin – RR). Таймер генерує переривання через певні інтервали часу. На час кожного переривання процес, що виконується в даний момент, поміщається в чергу готових до виконання процесів, і починає виконуватися черговий процес, вибраний відповідно до стратегії FCFS. Ця методика відома також як



а) квант часу, більший типового часу обслуговування

б) Квант часу, менший типового часу обслуговування

Рисунок 2 – Вплив довжини кванта часу на час відгуку

квантування часу, оскільки перед витісненням кожен процес отримує квант часу на виконання.

При круговому плануванні принциповим постає питання про тривалість кванта часу. Одне з емпіричних правил у цьому випадку звучить так: квант часу має бути трохи більшим, ніж час, потрібний для типового повного обслуговування. На рисунку 2 проілюстровано вплив тривалості кванта часу на час відгуку. В граничному випадку, коли квант часу перевищує час виконання найдовшого процесу, кругове планування вироджується в планування FCFS [3].

Щоб виявити один із основних недоліків даного алгоритму, розглянемо роботу з набором процесів, орієнтованих як на процесор, так і на операції вводу-виводу. Як правило, в процесів з інтенсивним вводом-виводом проміжок часу між двома операціями вводу-виводу, коли процес використовує процесор, менше, ніж в процесу, орієнтованого на використання процесора. В результаті можлива така ситуація: процес з інтенсивним вводом-виводом використовує процесор протягом короткого проміжку часу і опиняється в заблокованому стані в очікуванні завершення операції вводу-виводу. Після закінчення цієї операції він знову приєднується до черги готових до виконання процесів. З іншого боку, процес з інтенсивним використанням процесора зазвичай використовує відпущений йому квант часу повністю і миттєво повертається в чергу готових до виконання процесів. Отже, процес, орієнтований на роботу з процесором, отримує значно більший процесорний час, що спричиняє зниження продуктивності процесів з інтенсивним вводом-виводом, неефективне використання пристроїв вводу-виводу і збільшення часу відгуку [1].

Алгоритм вибору найкоротшого процесу (Shortest Job First) – це невитісняюча стратегія, при якій для виконання вибирається процес з найменшим очікуваним часом виконання. Основна складність алгоритму SJF полягає в тому, що для його реалізації необхідна оцінка часу виконання, потрібного кожному процесу. Основна проблема при використанні стратегії SJF полягає в тому, що при стабільній роботі коротких процесів і їх постійному надходженні час очікування та відгуку довгих процесів сильно зростає, що може спричинити їх "голодування". Крім того, використовувати даний алгоритм в системах з розподіленням часу або системах обробки транзакцій недоцільно через відсутність витіснення.

Стратегія найменшого часу, що залишається (Shortest Remaining Time), є витісняючою версією стратегії SJF. В цьому випадку диспетчер вибирає процес з найменшим очікуваним часом до закінчення процесу. При приєднанні нового процесу до черги процесів може виявитися, що його час виконання, що залишився, насправді менше, ніж час виконання процесу, що в даний момент виконується. Диспетчер, відповідно, може застосувати витіснення при готовності нового процесу. Як і при використанні стратегії SJF, планувальник для коректної роботи функції вибору повинен оцінювати час виконання процесу; в цьому випадку також є ризик "голодування" довгих процесів.

У випадку використання стратегії SRT немає таких великих перекосів на користь довгих процесів, як при використанні стратегії FCFS; на відміну від стратегії RR, тут не генеруються додаткові переривання, що знижують накладні витрати. Проте в цьому випадку відбувається збільшення накладних витрат через необхідність у фіксуванні і записі часу виконання процесів. У зв'язку з тим, що короткі завдання миттєво отримують перевагу над довготривалими завданнями, які виконуються, стратегія SRT має істотну перевагу над стратегією SJF за часом обігу та очікування [1]. Продемонструємо це на такому прикладі. Нехай до системи на виконання надходять 4 процеси (табл. 2).

Середній час очікування процесів для невитісняючого алгоритму SJF та витісняючого SRT показано на рис. 3.

При використанні алгоритму планування з пріоритетами кожному процесу присвоюється певне числове значення – пріоритет, відповідно до якого йому виділяється процесорний час. Як правило, менше числове значення пріоритету означає, що процес має більш високий пріоритет. Алгоритм пріоритетного планування може бути як витісняючим, так і невитісняючим. Процеси з однаковими пріоритетами плануються в порядку FCFS (рис. 4). Діаграма спрощена і ігнорує існування декількох черг заблокованих або призупинених процесів. Замість однієї черги готових до виконання процесів ми маємо

декілька таких черг, посортованих за зменшенням пріоритету: *RQ0*, *RQ1*, ..., *RQn*, тобто:

Пріоритет [RQi] > Пріоритет [RQj]
при $i < j$.

При виборі процесу диспетчер починає з черги процесів, яка має найвищим пріоритет (*RQ0*). Якщо в черзі знаходиться один або декілька процесів, для роботи вибирається процес з використанням певної стратегії планування. Якщо черга *RQ0* порожня, розглядається черга *RQ1* і т. д.

Таблиця 2 – Параметри процесів, що виконуються за стратегією SJF та SRT

Процес	Час прибуття	Час виконання
P_1	0	8
P_2	3	3
P_3	4	7
P_4	8	4

Рисунок 3 – Середній час очікування процесів для алгоритму SJF та SRT

Головна проблема пріоритетного планування полягає в тому, що при неналежному виборі механізму призначення і зміни пріоритетів низькопріоритетні процеси можуть не запускатися невизначено довгий час. Це буде відбуватися при постійному поступленні нових готових до виконання процесів з високим пріоритетом. Вирішення цієї проблеми може бути досягнуте за допомогою збільшення пріоритету процесу через певний час [3].

Для систем, у яких процеси можуть бути легко розсортовані по різних групах, був розроблений інший клас алгоритмів планування: багаторівневі черги зі зворотнім зв'язком. Його суть полягає в тому, що для кожної групи процесів створюється своя черга процесів, яка знаходяться в стані готовності. Цим чергам приписуються фіксовані пріоритети. Виконується витісняюче планування з використанням динамічного механізму.

При надходженні процесу в систему він поміщається в чергу *RQO* (рис. 5). Після першого виконання і повернення в стан готовності процес поміщається в чергу *RQ1*. Надалі при кожному витісненні цього процесу він вноситься до черги із все меншим пріоритетом. Відповідно, короткі процеси, що швидко виконуються, не можуть далеко зайти в ієрархії пріоритетів, тоді як довгі процеси поступово втрачають свій пріоритет. Таким чином, нові короткі процеси отримують перевагу у виконанні над старими довгими процесами [1].

Особливістю даного алгоритму є те, що всередині черг кожного рівня можуть застосовуватися різні алгоритми планування. Багаторівневі черги із зворотним зв'язком є найбільш загальним підходом до планування процесів. Вони найважчі в реалізації, але в той же час володіють найбільшою гнучкістю, оскільки для процесів з різними характеристиками застосовується найбільш вдалий алгоритм.

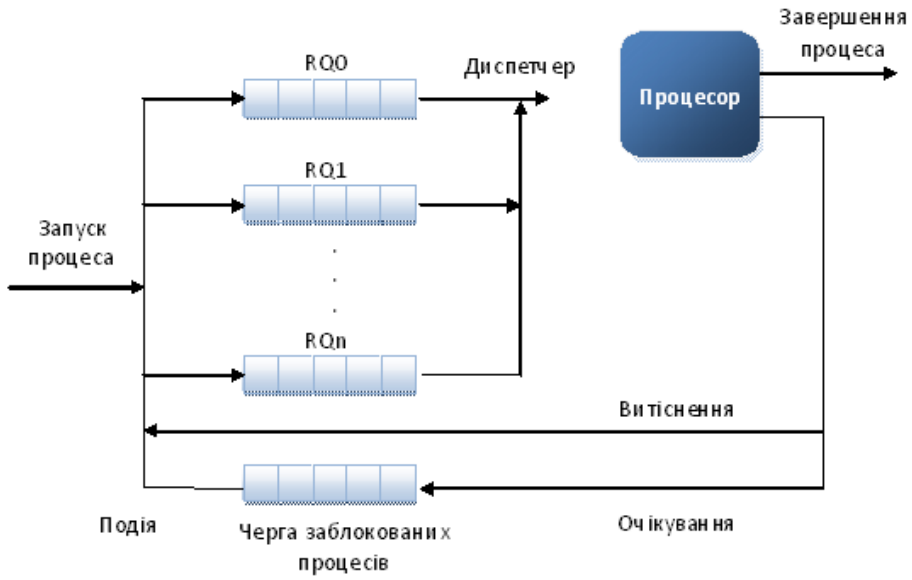


Рисунок 4 – Планування з використанням пріоритетів

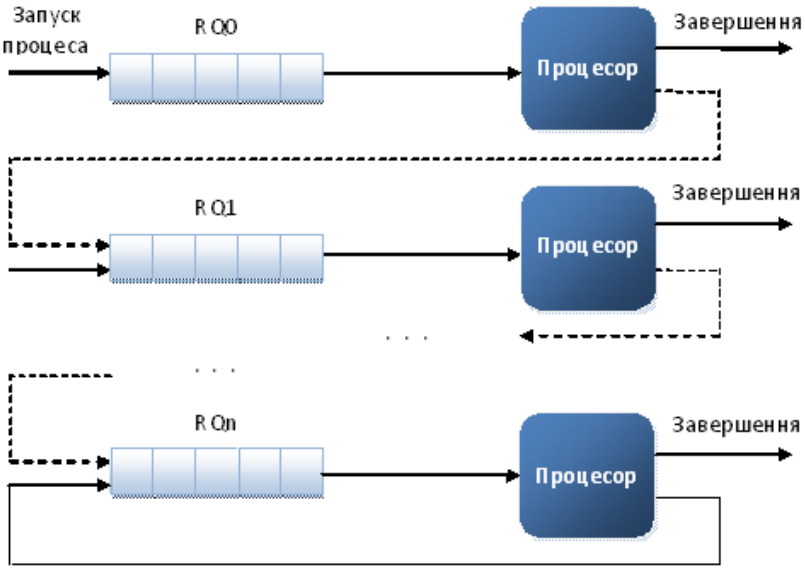


Рисунок 5 – Схема багаторівневої черги із зворотнім зв'язком

Незважаючи на значний інтерес до проблеми обслуговування процесів в комп'ютерних системах, особливо при розробці та реалізації операційних систем, проблемі імітаційного моделювання поведінки процесів у системі великої уваги не приділялось. Існуючі програмні продукти, які використовують MOSS, CPUSS, Cheddar, мають ряд недоліків: невелике число алгоритмів диспетчеризації, роботу яких можна зімітувати; неможливість налаштування користувачем параметрів цих алгоритмів; низька кількість статистичних показників, які характеризують роботу системи; нерозвинений і незручний графічний інтерфейс; проведення імітаційного експерименту тільки для однопроцесорної системи [4]. Тому доцільно розробити систему імітаційного моделювання поведінки процесів, яка б дозволяла проводити моделювання обробки процесів системою залежно від різної кількості процесорів та з врахуванням заданих алгоритмів планування, і тим самим визначити оптимальні характеристики та стратегії диспетчера системи для роботи із заданим набором процесів. Програма імітаційного моделювання повинна дозволяти проектувати комп'ютерну систему шляхом задання кількості процесорів та характеристик процесів, що надходять в систему на обробку, проводити імітаційне моделювання ініціалізації та обслуговування процесів з використанням заданих алгоритмів диспетчеризації, а також виводити статистичні показники та графічні результати моделювання [5]. Також необхідно розробити зручний користувацький інтерфейс програми, що дозволило б проводити дослідження та аналіз результатів без застосування додаткових засобів програмування. Програма моделювання процесів диспетчеризації в комп'ютерних системах написана мовою програмування C++ за допомогою крос-платформенного інструментарію розробки Qt версії 4.6.0. Це середовище зручне тим, що дозволяє запускати написане за його допомогою програмне забезпечення на більшості сучасних операційних систем шляхом простої компіляції тексту програми для кожної ОС без зміни початкового коду. Воно включає всі основні класи, які можуть бути потрібні при розробці прикладного програмного забезпечення, починаючи з елементів графічного інтерфейсу і закінчуючи класами для роботи з мережею, базами даних, і т. д. [6].

Програма дозволяє імітувати поведінку системи масового обслуговування, а саме комп'ютерної системи із заданою кількістю обслуговуючих пристроїв (процесорів), що обробляє деякий набір задач (процесів), згідно вказаних алгоритмів диспетчеризації. Процеси, що поступають в систему на виконання, характеризуються певними властивостями, насамперед швидкістю надходження в систему та середнім часом обслуговування. Статистичні показники, отримані в результаті імітаційного експерименту, дають змогу оцінити вплив різних стратегій планування на виконання процесів системи.

В основу проведення експерименту покладемо систему із одним процесором, на обслуговування до якого надходять 1000 процесів із середньою швидкістю 20 процесів/секунду та середнім часом обслуговування 45 мс. Таким чином, коефіцієнт використання процесора дорівнює 90%. Вважаємо, що інтервали між надходженнями процесів в систему та час обслуговування розподілені експоненціально. Експеримент проводиться як для системи з коефіцієнтом завантаження 90% (один процесор), так і для системи з коефіцієнтом 45% (2 процесори) та 30% (3 процесори). Причому для багатопроцесорних систем розглядається два варіанти організації черг: спільна черга до всіх процесорів та окрема черга до кожного процесора.

Критеріями оцінки даного експерименту слугують:

Таблиця 3 – Результати експерименту для алгоритму «першим прийшов – першим обслужений»

Коефіцієнт завантаження, %	Схема системи	Середній час обігу, мс	Середній час очікування, мс	Нормалізований час обігу
90	M/M/1	412	366	33
45	M/M/2	53	9	1.8
	2 M/M/1	71	24	2.9
30	M/M/3	46	1	1.1
	3 M/M/1	53	7	1.9

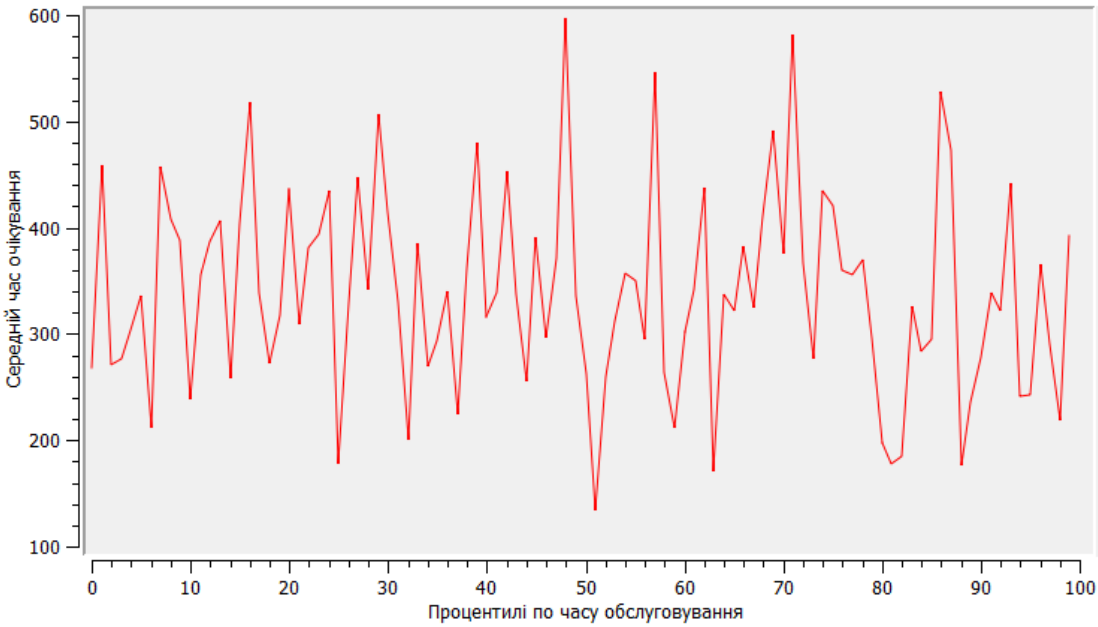


Рисунок 6 – Залежність часу очікування від часу обслуговування процесів для алгоритму «першим прийшов – першим обслужений»

– середній час обігу (повний час, який процес проводить у системі);

– середній час очікування (час, протягом якого процес чекає на виконання в черзі);

– середній нормалізований час обігу, що вказує відносну затримку процесу в системі і визначається як відношення часу обігу до часу виконання. Мінімальне значення цього відношення дорівнює 1. Збільшення цього значення відповідає зниженню рівня обслуговування.

Зважаючи на випадковий розподіл вхідних характеристик процесів, для одержання сталих результатів доцільно провести декілька досліджень із заданим набором параметрів системи.

Першим було проведено експеримент, у якому поведінка процесів системи визначалась алгоритмом «першим прийшов – першим обслужений». Моделювання здійснювалось 10 разів. Отримані результати подані в таблиці 3.

Для однопроцесорної системи продуктивність алгоритму «першим прийшов – першим обслужений» дуже низька, адже середній час обігу процесу в системі майже в 10 разів перевищує час обслуговування. При цьому короткі процеси страждають найбільше – їх відносна затримка в системі досягає 300 одиниць. З іншого боку, хоча час очікування достатньо великий, проте він практично однаковий як для коротких, так і для довготривалих процесів (рис. 6).

При збільшенні кількості процесорів продуктивність алгоритму значно підвищується і процеси майже не затримуються в черзі, а відразу ж виконуються. Причому система із спільною чергою до декількох процесорів працює набагато ефективніше, ніж система із окремими чергами до кожного процесора.

При дослідженні системи із стратегією кругового планування додатково визначався вплив довжини кванта часу на роботу заданої стратегії. Розглядалося такі три випадки:

Квант часу менший від середнього часу обслуговування – 30 мс;

Квант часу дорівнює середньому часу обслуговування – 45 мс;

Квант часу більший за середній час обслуговування – 60 мс.

Результати експерименту наведено в таблиці 4. Як видно із одержаних результатів, оптимальна величина кванта часу, при якій алгоритм циклічної диспетчеризації працює найефективніше, становить 45 мс, тобто дорівнює середньому часу обслуговування процесу. Для двопроцесорної системи вибір кванта часу вже не відіграє такого вагомого значення, оскільки процеси дуже швидко обслуговуються.

Таблиця 4 – Результати експерименту для алгоритму кругового планування

Коефіцієнт завантаження, %	Квант часу, мс	Середній час обігу, мс	Середній час очікування, мс	Нормалізований час обігу
90	30	457	415	21
	45	421	368	19
	60	449	403	26
45	30	60	14	1.4
	45	58	12	1.4
	60	57	12	1.6

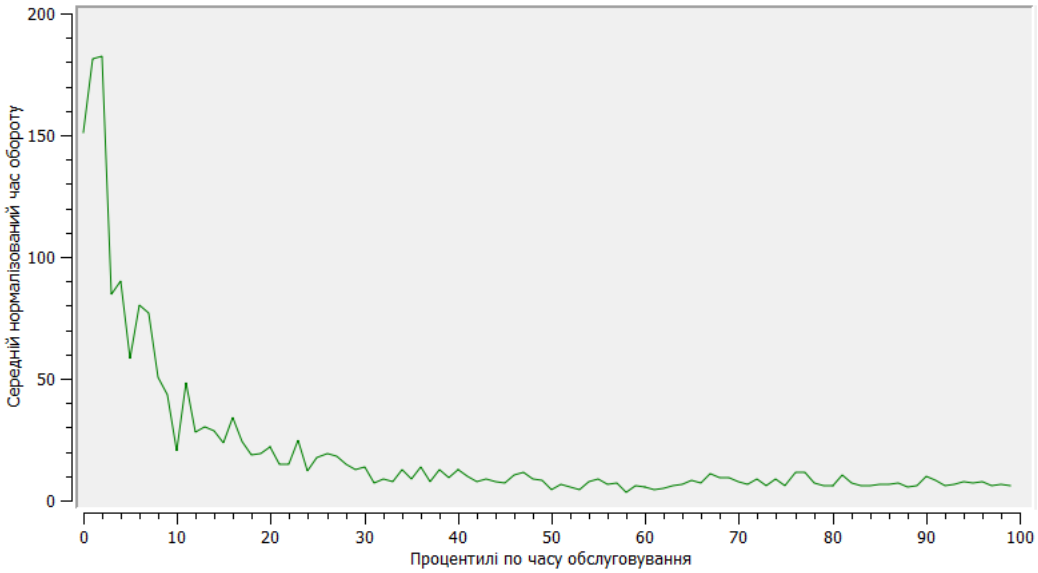


Рисунок 7 – Залежність нормалізованого часу обігу від часу обслуговування процесів для стратегії кругового планування

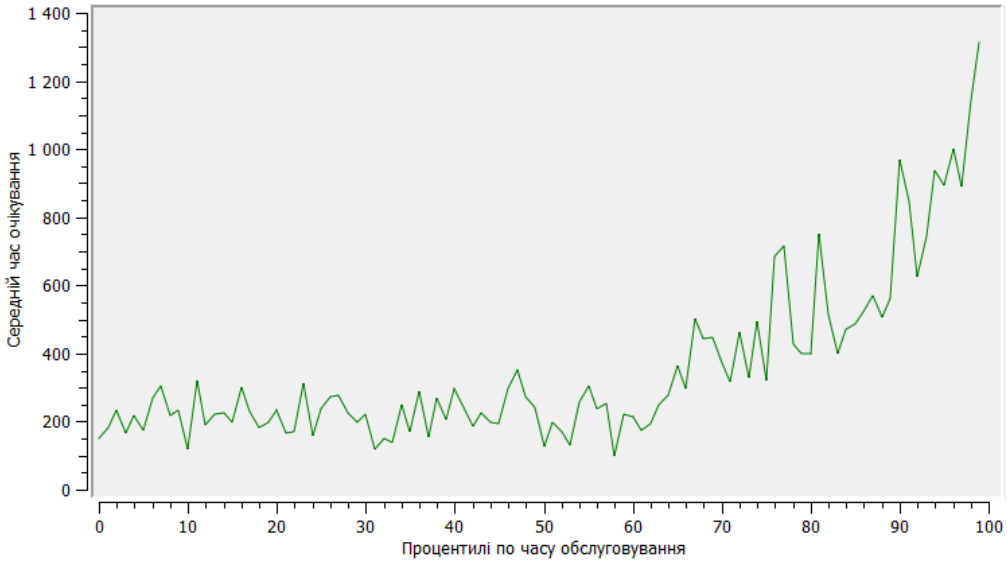


Рисунок 8 – Залежність часу очікування від часу обслуговування процесів для алгоритму кругового планування

За винятком найкоротших процесів, час обслуговування яких менший за один квант часу, при цій стратегії планування нормалізований час обігу (рис. 7) становить приблизно 10 одиниць для всіх процесів, таким чином забезпечуючи безпристрасність.

Проте внаслідок цієї безпристрасності час обігу довгих процесів різко розтягується і відповідно їх час очікування в черзі збільшується в рази (рис. 8).

У алгоритмі вибору найкоротшого процесу дослідження проводилось для витісняючого і

Таблиця 5 – Результати експерименту для алгоритму вибору найкоротшого процесу

Коефіцієнт завантаження, %	Витіснення	Середній час обігу, мс	Середній час очікування, мс	Нормалізований час обігу
90	ні	190	144	4.7
	так	160	114	1.7
45	ні	54	9	1.4
	так	50	6	1

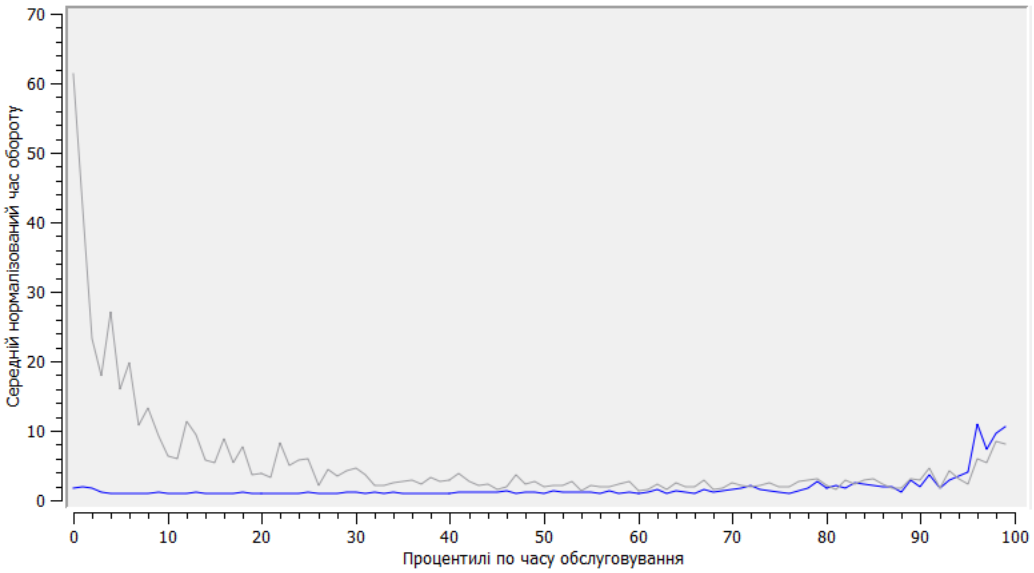


Рисунок 9 – Залежність нормалізованого часу обігу від часу обслуговування процесів для алгоритму вибору найкоротшого процесу

Таблиця 6 – Результати експерименту для алгоритму багаторівневої черги із зворотнім зв'язком

Коефіцієнт завантаження, %	Квант часу, мс	Середній час обігу, мс	Середній час очікування, мс	Нормалізований час обігу
90	30	417	372	5.6
	45	410	365	6.4
45	30	59	13	1.1
	45	58	13	1.1

невитісняючого варіантів. Результати моделювання представлені в таблиці 5.

Продуктивність при використанні витісняючого варіанту алгоритму (алгоритм вибору найменшого часу, що залишився) є вищою, ніж при використанні невитісняючої стратегії. На рис. 9 показано графік залежності середнього нормалізованого часу обігу від процентилів по часу обслуговування для обох варіантів алгоритму (світлішим позначено невитісняючу стратегію, темнішим – витісняючу). Як видно з рисунка, невитісняючий алгоритм краще обслуговує 10% найдовших процесів, на відміну від витісняючого, що краще справляється з короткими процесами.

При моделюванні стратегії багаторівневих черг із зворотнім зв'язком, кількість рівнів черг дорівнювала 3 (достатнє число для того, щоб довгі процеси переходили на нижчі рівні), а процеси вважалися такими, що не мають пріоритетів (при проведенні даного експерименту вважається, що всі процеси рівноправні). Довжина кванта часу була вибрана 45 мс згідно проведеного дослідження для алгоритму кругового планування. Але, зважаючи на те, що з переходом на кожен нижчий рівень черги процес отримував у своє розпорядження процесор протягом двох квантів часу попереднього рівня, було цікаво прослідкувати вплив короткого кванта часу на продуктивність алгоритму. Отримані результати дослідження подано в таблиці 6.

Як виявилось, при такій стратегії диспетчеризації квант часу не має настільки вагомого впливу на продуктивність, як у випадку кругового планування, адже у кожному наступному підході до процесора йому надається вдвічі більший час для виконання.

Алгоритм багаторівневої черги із зворотнім зв'язком достатньо непогано працює із ко-

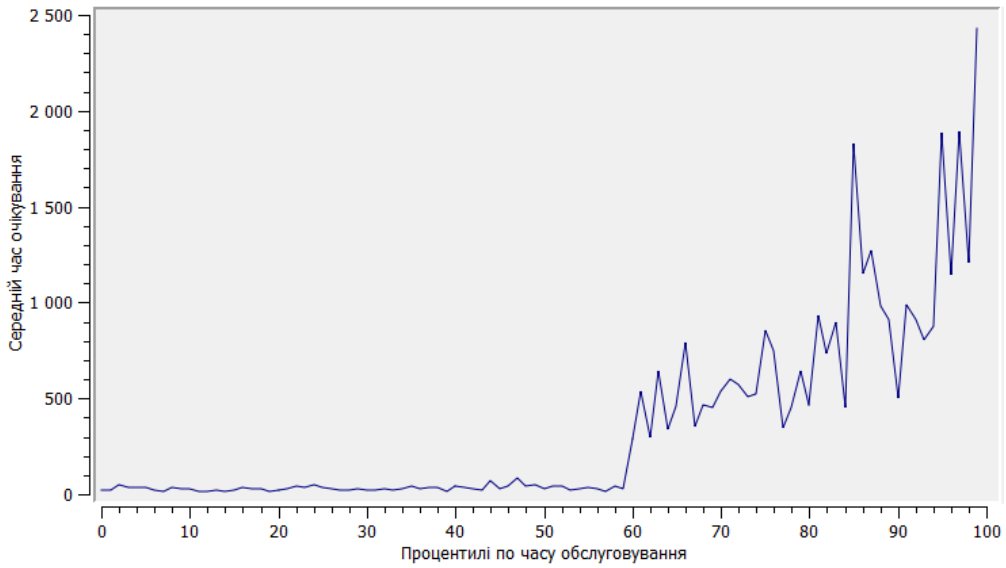


Рисунок 10 – Залежність часу очікування від часу обслуговування процесів для алгоритму багаторівневої черги із зворотнім зв'язком

Таблиця 7 – Результати експерименту для всіх алгоритмів диспетчеризації

	FCFS	RR	SJF	PP	MFQ
Коефіцієнт завантаження ρ = 90%					
Середній час обігу, мс	622	653	235	657	650
Середній час очікування, мс	563	593	176	598	591
Нормалізований час обігу	39	27	2	32	7
Коефіцієнт завантаження ρ = 45%					
Середній час обігу, мс	71	71	66	72	71
Середній час очікування, мс	11	11	6	12	11
Нормалізований час обігу	1.6	1.3	1	1.3	1.2
Коефіцієнт завантаження ρ = 30%					
Середній час обігу, мс	62	62	62	62	62
Середній час очікування, мс	1.3	1.5	1.2	1.5	1.5
Нормалізований час обігу	1	1	1	1	1

роткими процесами (рис. 10), проте час очікування близько 40% найдовших процесів дуже великий.

Для отримання узагальнених результатів по всіх алгоритмах диспетчеризації було проведено ще один експеримент. В систему на обслуговування надходило 1000 процесів із середньою швидкістю 15 процесів/секунду та середнім часом обслуговування 60 мс. Інтервали часу між надходженнями процесів в систему та час обслуговування розподілені експоненціально. Для алгоритмів кругового планування та багаторівневої черги із зворотнім зв'язком квант часу задали рівним середньому часу обслуговування. Алгоритм вибору найкоротшого процесу працює в ефективнішому витісняючому варіанті, а для алгоритму пріоритетного планування задано 3 групи пріоритетів.

Проведено три досліди: в першому коефіцієнт завантаження процесора дорівнював 90%, у другому – 45%, в третьому – 30%, причому в останніх двох випадках використовується спільна черга до процесорів. Відмова від проведення моделювання системи з окремою чергою до кожного із процесорів викликана тим, що в проведених експериментах даний підхід був неефективним і лише знижував вихідні показники системи для всіх алгоритмів диспетчеризації. Отримані в результаті проведення експерименту дані занесені до загальної таблиці 7.

Як видно, для заданого набору процесів алгоритм SJF, а точніше його витісняючий варіант, є оптимальним з погляду мінімізації середнього часу обігу та середнього часу очікування процесів серед усіх алгоритмів диспетчеризації (табл. 7). Відносна затримка процесу в системі (середній нормалізований час обігу) для даного алгоритму також є найнижчою (табл. 7). Оскільки алгоритм SJF надає перевагу коротким процесам, час очікування приблизно 5% найдовших процесів сильно розтягується (рис. 11), і може викликати "голодування" цих процесів.

Подібно до цього діє й алгоритм багаторівневої черги із зворотнім зв'язком (MFQ). Попри низьку відносну затримку процесу в систе-

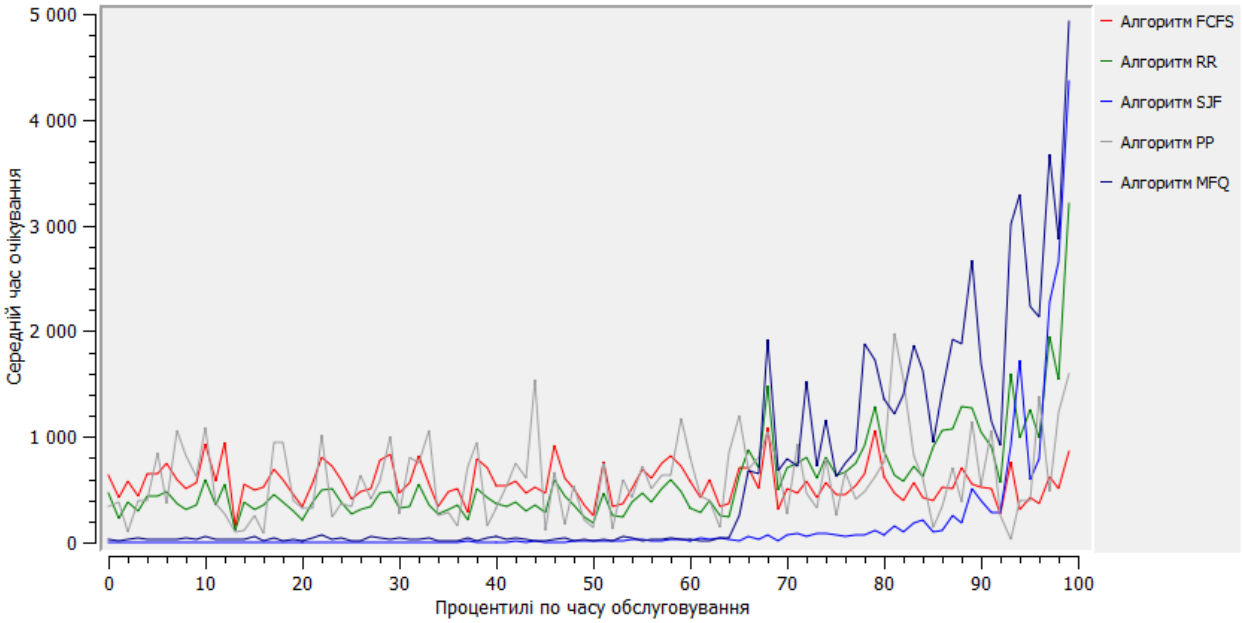


Рисунок 11 – Залежність часу очікування від часу обслуговування процесів для різних алгоритмів диспетчеризації при 90% завантаженні процесора

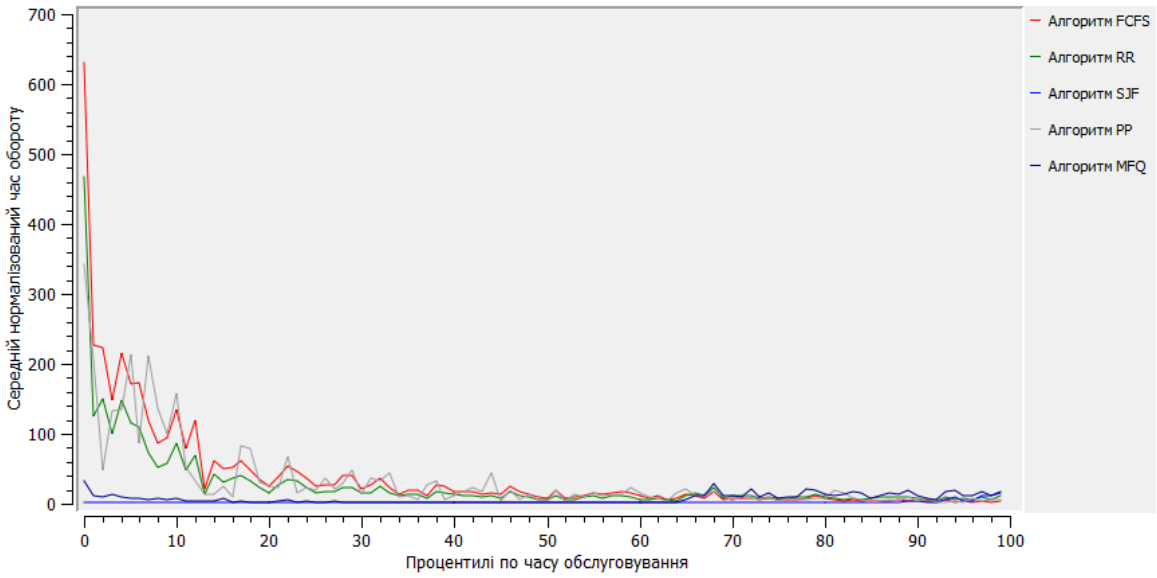


Рисунок 12 – Залежність нормалізованого часу обігу від часу обслуговування процесів для різних алгоритмів диспетчеризації при 90% завантаженні процесора

ні (рис. 12), час очікування приблизно 35% найдовших процесів є найбільшим серед усіх алгоритмів (рис. 11), що дає підставу говорити про можливе виникнення "голодування".

На відміну від двох попередніх алгоритмів, алгоритм «першим прийшов – першим обслужений» (FCFS) віддає перевагу довгим процесам. Хоча середній нормалізований час обігу процесів є найбільшим для даного алгоритму (табл. 7), затримка близько 10% найдовших процесів у системі є найменшою серед усіх алгоритмів (рис. 12) Ще однією особливістю, яка вирізняє цей алгоритм, є середній час очікування, який практично однаковий як для коротких, так і для довгих процесів (рис. 11).

Подібно до FCFS працює й алгоритм кругового планування (RR) (табл. 7), проте він явно ефективніше справляється з короткими процесами. Щодо довготривалих процесів, то середній час очікування приблизно 35% найдовших процесів для даного алгоритму більший за час очікування цих же ж процесів у стратегії FCFS (рис. 12).

Алгоритм витісняючого пріоритетного планування (PP) має подібні статистичні показники до алгоритмів FCFS та RR (табл. 7). Поведінка цього алгоритму значною мірою визначається порядком надходження процесів з вищим пріоритетом, а тому його вплив на короткі та довгі процеси може бути різним в залежності від цього фактору, що і демонструють рис. 11 та рис. 12.

На рис. 13 зображено динаміку росту черги в часі. Видно, що при використанні витісняю-

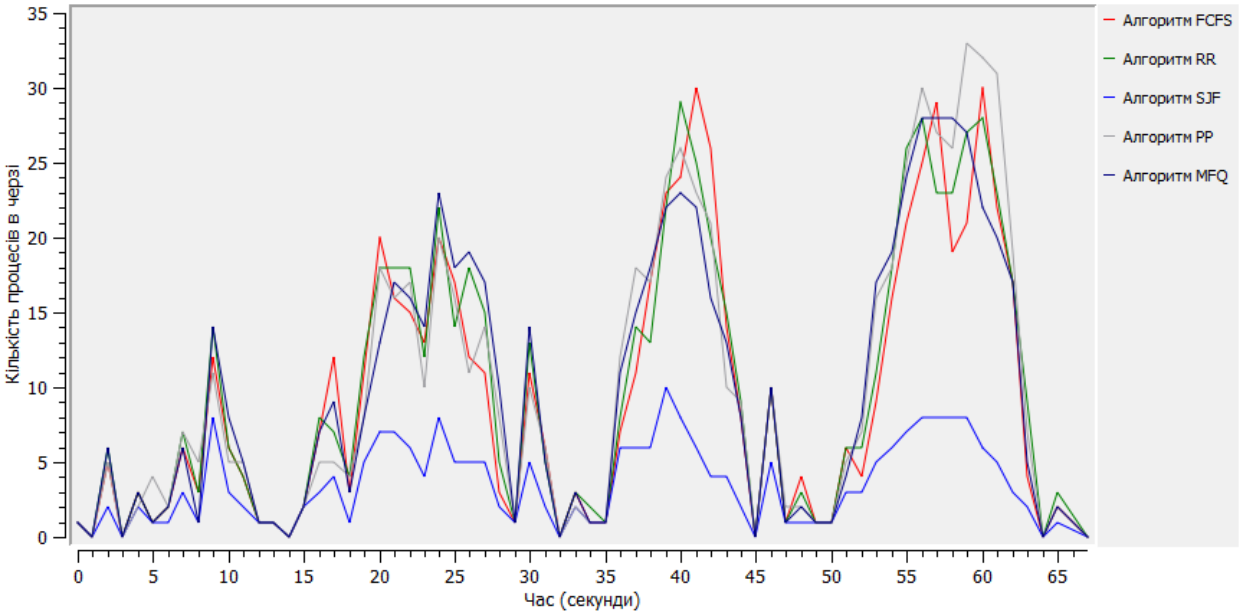


Рисунок 13 – Динаміка росту черги в часі при використанні різних стратегій планування

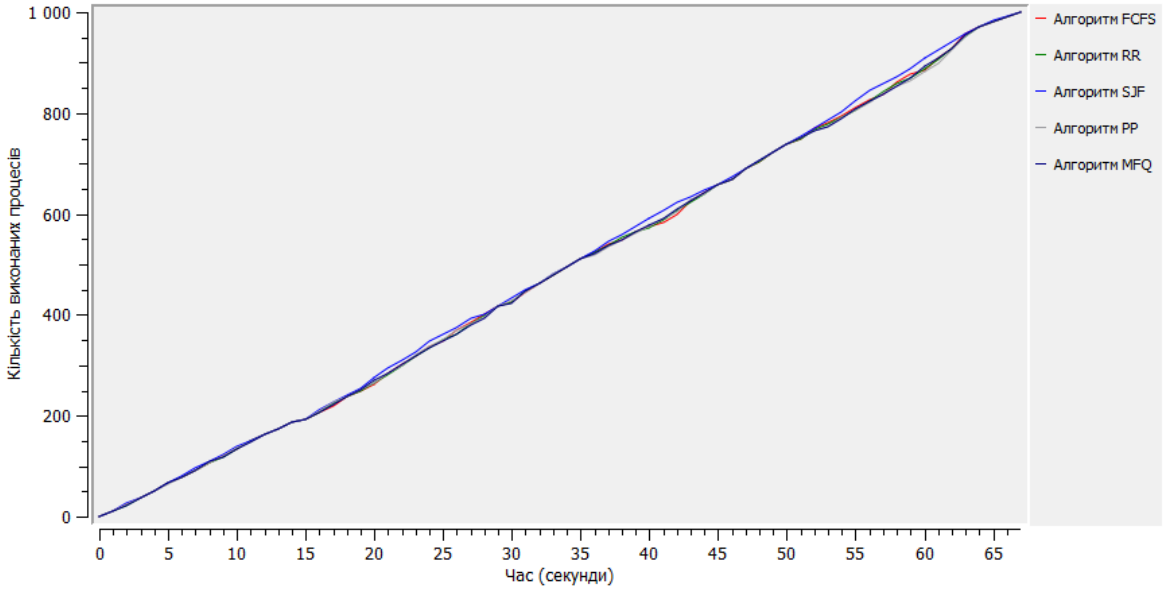


Рисунок 14 – Зміна кількості виконаних процесів в часі при використанні різних стратегій планування при 90% завантаженості процесора

чої стратегії SJF середня кількість процесів, що очікують виконання, є найменшою серед всіх алгоритмів. Це зумовлено тим, що короткі процеси виконуються відразу, і в черзі залишаються лише довгі процеси, на відміну від решти алгоритмів, де короткі процеси змушені чекати закінчення виконання довготривалих.

Цікавою є також зміна кількості виконаних процесів у часі для кожного із алгоритмів диспетчеризації. З рисунку 14 видно, що всі стратегії планування працюють практично рівномірно і завершують процеси в однакові терміни.

Зміни даної картини можна домогтися лише при перевантаженні системи, скажімо збільшенням вхідної швидкості надходження процесів вдвічі, при якому коефіцієнт використання процесора стане дорівнювати 100%, а черга постійно зростатиме, допоки всі процеси не надійдуть до системи (рис. 15).

У цьому випадку алгоритм вибору найкоротшого процесу (SJF) завершує процеси швидше від інших алгоритмів до моменту, коли черга досягне свого максимуму. Очевидно, що таке збільшення пропускної здатності, пов'язано з тим, що дана стратегія завершує спочатку коротші процеси з черги.

При дослідженні двопроцесорної системи з коефіцієнтом завантаженості 45%, помітно, що вплив алгоритмів диспетчеризації на продуктивність значно зменшився (таблиці 4, 5). Сама ж продуктивність системи зросла в рази. Так, наприклад, для алгоритму FCFS середній час обігу процесу зменшився у 9 разів, середній час очікування у 51 раз, а нормалізований час обігу

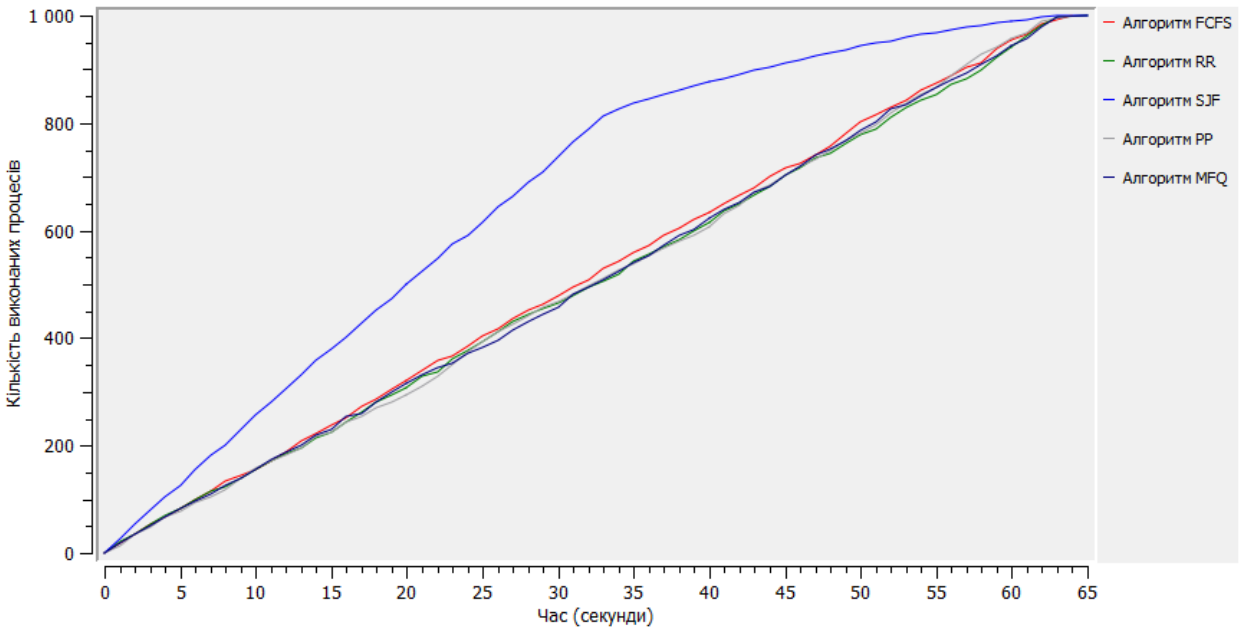


Рисунок 15 – Зміна кількості виконаних процесів у часі при використанні різних стратегій планування при 100% завантаженості процесора

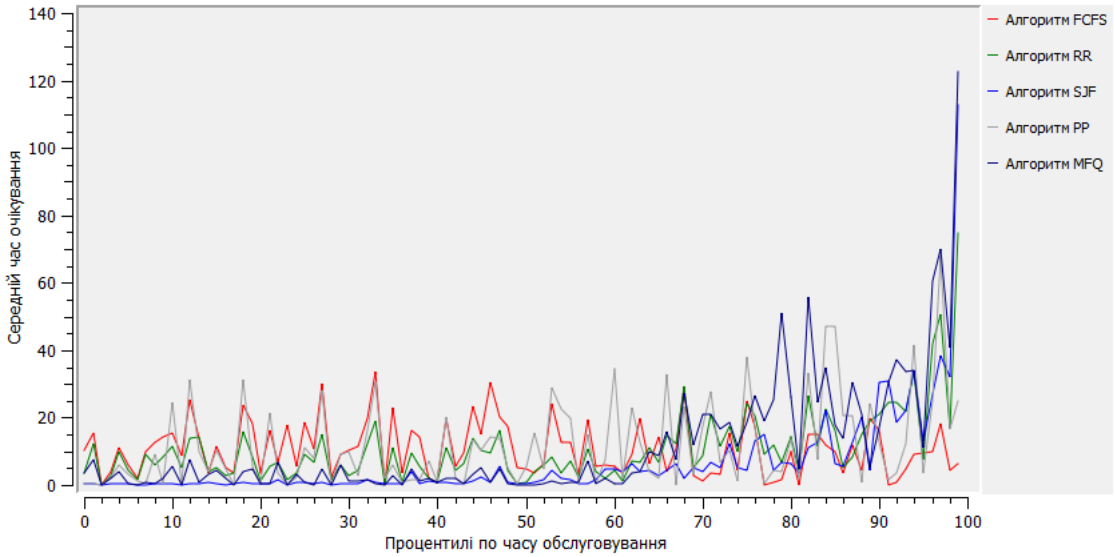


Рисунок 16 – Залежність часу очікування від часу обслуговування процесів для різних алгоритмів диспетчеризації при 45% завантаженні процесора

став практично рівним 1. Всі стратегії в даному випадку працюють майже однаково ефективно по всіх критеріях. Виняток складає середній час очікування приблизно 5% найдовших процесів, для яких оптимальним залишається алгоритм «першим прийшов – першим обслужений» (рис. 16).

Відносна затримка процесів у двопроцесорній системі значно перевищує середнє значення лише для 2% найкоротших процесів для всіх алгоритмів, за винятком SJF (рис. 17).

При моделюванні аналогічної системи з коефіцієнтом завантаженості 30% (три процесори) вплив стратегій планування на систему нівелюється. Для всіх алгоритмів середній час обігу стає рівним середньому часу обслуговування, час очікування становить близько 1 мс, і відповідно процеси, поступаючи в систему, відразу ж виконуються.

IV Висновок. В ході проведення імітаційного експерименту було зроблено порівняльний аналіз продуктивності системи для заданих стратегій диспетчеризації з точки зору оптимізації вибраних системних критеріїв, таких як середній час обігу процесу, середній час очікування та середній нормалізований час обігу, з якого слідує, що не існує універсального алгоритму диспетчеризації, тому при застосуванні будь-якого з них необхідно враховувати параметри процесів, з якими вони працюють, та критерії результату, що має бути досягнутим. Дослідження ж впливу алгоритмів планування на багатопроцесорні системи виявило, що з ро-

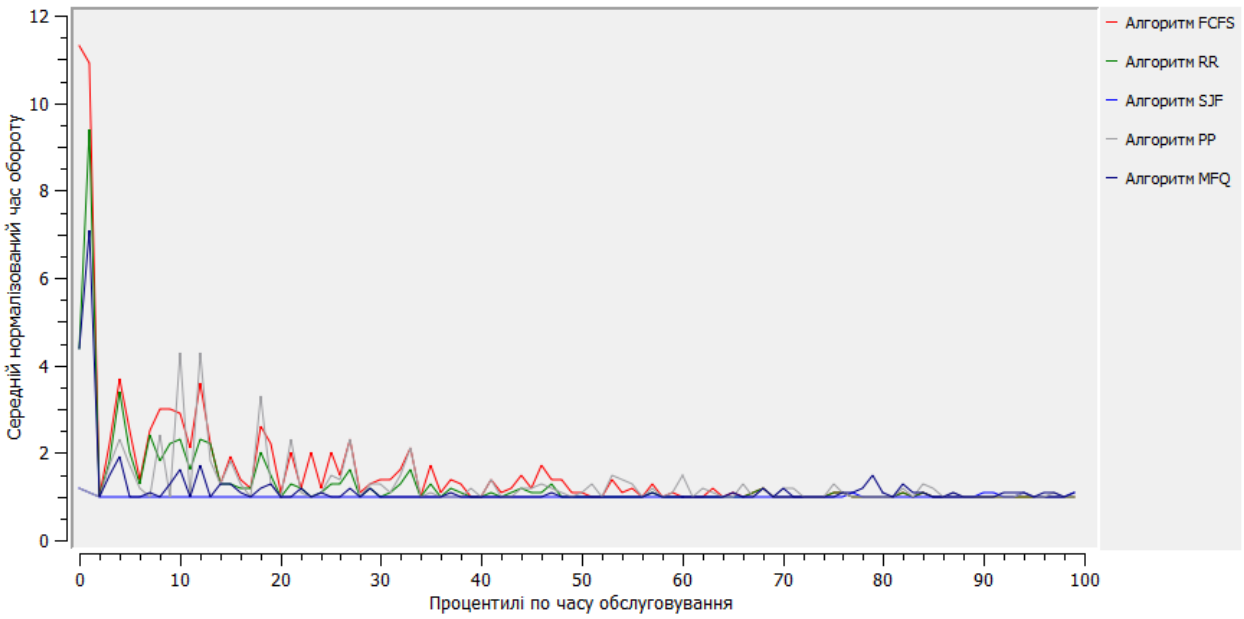


Рисунок 17 – Залежність нормалізованого часу обігу від часу обслуговування процесів для різних алгоритмів диспетчеризації при 45% завантаженні процесора

стом кількості процесорів різниця в їх продуктивності та вплив на роботу системи різко знижується.

Література

1 Crowley C. Operating Systems: A Design-Oriented Approach. — Chicago: Irwin, 1997. – 579 p.

2 Столлингс В. Операционные системы. 4-е издание. – М.: Издательский дом «Вильямс», 2004. – 848 с.: ил.

3 Таненбаум Э. Операционные системы. Разработка и реализация. Классика CS / Э. Таненбаум, А. Вудхалл. 3-е изд. — СПб.: Питер, 2007. – 704 с: ил.

4 Averill M., Law W., David Kelton. Simulation Modeling and Analysis. – Mc GrawHill, 2000. – 635 p.

5 Строгалев В.П. Имитационное моделирование / В.П. Строгалев, Толкачева. – М.: МГТУ им. Баумана, 2008. – 737с.

6 Джосьютис Н. С++ стандартная библиотека для профессионалов. – СПб: Питер, 2004. – 730 с.

Стаття надійшла до редакційної колегії 12.05.11
Рекомендована до друку професором ***Горбійчуком М.І.***