УДК 519.687
DOI: 10.15587/2313-8416.2015.46591

МЕТОДИ УПРАВЛІННЯ РЕСУРСАМИ І ДОДАТКАМИ В ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ НА БАЗІ ХМАРНИХ ТЕХНОЛОГІЙ

© К. А. Мацуєва

В даній статті описано методи керування ресурсами і додатками, що знаходяться в складі інформаційної системи для наукових досліджень (ІСНД). Представлено модель керування потоками запитів в ІСНД і приведено результати досліджень реальної хмарної системи з використанням додаткового модуля розподілення навантаження, запрограмованого на мові Python
Ключові слова: *інформаційна система, хмарні обчислення, віртуальні ресурси, розподіл навантаження, керування ресурсами*

This article describes the methods of resource management and applications that are parts of an information system for science research (ISSR). The control model of requests in ISSR is given and results of working real cloud system using the additional module of load distribution programmed in Python are presented
Keywords: *information system, cloud computing, virtual resources, load distribution, resource management*

1. Вступ

В даний час існує безліч рішень, що дозволяють здійснювати управління розподілом навантаження між обчислювальними ресурсами. Серед них найбільш перспективним є концепція хмарних обчислень. Ця технологія дозволяє уніфікувати доступ до ресурсів, що дуже важливо для систем, що вимагають високої якості обслуговування і забезпечують доступність послуг в цілодобовому режимі [1]. Використання концепції хмарних обчислень дозволяє забезпечити збереження і обслуговування значних об'ємів даних, використовуваних системою [2, 3].

2. Постановка проблеми

Досить ефективним вважається прогнозування поведінки клієнтів використовуючи механізми попередньої реєстрації і статистику споживання ресурсів для кожної з підсистем [4]. Таким чином, можна визначити об'єм потужностей для обслуговування потоку заявок, що надходять.

Потоки запитів (трафік) до мультимедійних додатків, як правило, відрізняються від моделі простого пуассонівського потоку, що описується експоненційною функцією розподілу інтервалу часу між моментами надходження заявок. При цьому кожен з джерел запитів, що бере участь у процесі створення потоку пакетів, володіє власними значеннями питомої інтенсивності навантаження. У кожний момент часу інтенсивність навантаження результуючого потоку запитів користувачів залежить від того, до яких додатків вони спрямовані, яким каналом (віртуальним ресурсом) вони обслуговуються, а також яке співвідношення їх чисельності для різних додатків.

3. Огляд літератури

Як правило, завдання планування вирішуються шляхом їх зведення до завдань комбінаторної оптимізації, зокрема, до завдань лінійного програмування, пошуку максимального потоку та ін. Також нерідко застосовується динамічне програмування або метод гілок і меж [5]. Слід враховувати, що завдання складання плану NP-повне, отже, складання оптимальних за заданим критерієм розкладів вимагає експоненціального часу, що неприпустимо для використання в обчислювальних системах, так як кожен цикл планування займає значний проміжок часу. Для вирішення даної задачі застосовують евристичні алгоритми планування, що будують субоптимальний розклад за прийнятний поліноміальний час.

Розглянемо алгоритми планування, що можуть бути використані для систем з архітектурою хмарних обчислень. Одним з найпростіших методів планування, що застосовуються для складання розкладу, є алгоритм First Come First Served (FCFS) [6, 7]. У кожному циклі планування, з черги виділяються запити і призначаються на визначені обчислювальні вузли. При цьому запити в черзі впорядковані згідно часу їх надходження. Крім описаного алгоритму застосовують і інші, що використовують список в якості основного елемента складання плану, включаючи Shortest Job First (найкоротша задача перша), Random Job First (випадкова задача перша) та ін. [7]. Істотний недолік спискових алгоритмів – низька завантаженість обчислювальних вузлів в силу наявності великої кількості вікон у створюваних розкладах, що призводить до простоювання та неефективного використання обчислювальних ресурсів.

Для вирішення цих проблем Аргонською національною лабораторією запропонований агресивний варіант алгоритму Backfill (алгоритму зворотного заповнення). Він переслідує дві конфліктуючі цілі - підвищення ефективності використання обчислювальних ресурсів шляхом заповнення порожніх вікон у розкладі та запобігання утримання заявок у черзі на обслуговування обчислювальним вузлом за рахунок механізму резервування. При цьому запити, що очікують, зберігаються в черзі і впорядковані відповідно до пріоритетів. У кожному циклі планування обчислювальні ресурси виділяються згідно встановленим пріоритетам. Особливістю алгоритму є те, що заявка, що надійшла на обслуговування, не зможе отримати доступ до обчислювальних ресурсів, поки вони не відведені пріоритетним запитам. Таким чином, ресурси обчислювальної системи в першу чергу виділя-

ються пріоритетним заявкам, а решта заповнюють вікна, що утворилися в ході резервування в розкладі в порядку, встановленому пріоритетами.

У дослідженнях Д. Фейтельсон і А. Вейл з Єврейського університету м. Єрусалиму запропонований консервативний варіант алгоритму Backfill відмітною особливістю якого є більш жорстка залежність від пріоритетів. Запити, що надходять, не обслуговуються, поки в розкладі не виділені ресурси для більш пріоритетних заявок.

Дослідження показали, що консервативний варіант алгоритму Backfill по завантаженості обчислювальних вузлів не поступається агресивному варіанту, і, на відміну від останнього, дозволяє робити точні припущення про час запуску кожного завдання, що знаходиться в черзі [8]. Інша не менш важлива перевага консервативного варіанту алгоритму Backfill - можливість точного резервування ресурсів для запиту відразу ж після надходження в чергу. Це дає можливість застосовувати алгоритм Backfill в хмарних системах. Для роботи алгоритму Backfill суттєвою є наявність оцінок часу виконання запитів, що надходять в чергу. Однак при великій неточності в оцінках ефективність роботи алгоритму падає, а також знижується середня завантаженість обчислювальних вузлів.

Більшість широко відомих методів складання розкладів і вибору заявок з черги запитів не враховують топологію, використовувану в обчислювальній системі. Так, при роботі алгоритму First Fit (перший, що надходить) вибираються перші підходящі вузли з першого кращого вікна вільних обчислювальних вузлів. В алгоритмі Best Fit (найкращий, що надходить) - спочатку проглядаються підходящі вікна з кінцевим часовим інтервалом, і серед них вибирається те, закриття якого дасть найменшу залишкову площу. Якщо немає підходящих кінцевих вікон, то вибирається відповідне нескінченне вікно, найбільш близьке по необхідним ресурсам для виконання запиту. Такий підхід часто застосовують спільно з алгоритмом зворотного заповнення (Backfill). Вибір згідно з алгоритмом Fastest Node First (найшвидший вузол перший) здійснюється з найбільш продуктивних вузлів у відповідних вікнах розкладу. Крім перерахованих алгоритмів, так само застосовують пошук вузлів згідно з алгоритмом Least Utilized Node First (найменш завантажений вузол перший). Він дозволяє вибрати найменш завантажені відповідні вузли в доступному для вибору вікні. Відповідний обчислювальний вузол для обслуговування запитів, що надходять може бути обраний випадковим чином з використанням алгоритму Random First (RF, випадковий вузол перший) [9, 10].

Розглянемо алгоритми планування для хмарних систем, які є окремим випадком грід-систем з різними схемами роботи. Більшість сучасних хмарних систем використовують централізовану схему планування, яка передбачає застосування одного з двох типів алгоритмів планування – «жадібні» або «ліниві».

У випадку «жадібних» алгоритмів глобальна черга задач не використовується. Для кожної задачі, що формується контролером хмарної системи на основі вхідних запитів користувачів, за деяким принципом обирається відповідний обчислювальний вузол, куди передається задача. Вибір відповідного обчислювального ресурсу здійснюється за допомогою інформаційної бази, яка містить актуальні відомості про поточний стан і конфігурації обчислювальних ресурсів. Збір подібних відомостей здійснюється спеціалізованими розподіленими системами моніторингу. При цьому вибирається ресурс з мінімальним значенням деякої інтегральної характеристики, наприклад, коефіцієнта використання, довжини черги завдань та інших. Проте, використання подібних алгоритмів потенційно призводить до зависання завдань і неефективної та незбалансованої завантаженості обчислювальних ресурсів.

У випадку «лінивих» алгоритмів планування запити користувачів поміщаються в глобальну чергу. Локальний планувальник обчислювального ресурсу запитує нові дані у контролера в певні моменти часу, обумовлені адміністративною політикою. У цьому випадку також знижується ефективність роботи системи, тому локальні планувальники обчислювальних вузлів не володіють достатньою інформацією для складання повних і ефективних розкладів.

Таким чином, з викладеного вище випливає, що для розробки ефективних алгоритмів керування ресурсами в хмарній системі необхідно отримати відомості про особливості роботи сервісів, що розміщені на обчислювальних вузлах. Це дозволить більш чітко визначити і локалізувати потенційні проблемні ділянки.

Завдання управління ресурсами виникає при конкуренції за ресурси, зокрема при одночасному використанні віртуальних машин на одному обчислювальному вузлі. Основним завданням при цьому є попереднє планування виділення ресурсу і динамічний перерозподіл ресурсів при інтенсивному надходженні запитів користувачів. У технології хмарних обчислень передбачені механізми для вирішення перерахованих особливостей. Однак вони не враховують різнорідність використовуваних додатків і сервісів, що достатньо критично для сучасних інформаційних систем. Отже, потрібна розробка нових алгоритмів планування для забезпечення якості обслуговування користувачів інформаційних систем.

4. Проведення експериментального дослідження методів керування віртуальними ресурсами і додатками

Використовуючи різні оптимізаційні алгоритми на кожному з етапів можна вплинути на структуру трафіку, забезпечивши тим самим якість обслуговування для виділеного сервісу. При цьому вхідні потоки зазнають значних змін і в підсумковому трафіку з'являються довгострокові залежності в інтенсивності запитів, що дозволяє ввести зворотний зв'язок для компонентів обслуговуючої системи 11, 12]. Відмінною особливістю імітаційної моделі, побудованої для дослідження процесу обслуговування запитів, є набір ознак, що характеризують кожну з заявок [13]:

• ресурсомісткість – оцінюється з використанням рейтингу затребуваності основних ресурсів системи;

• передбачуваний час виконання – оцінюється з використанням статистики обслуговування однотипних заявок в залежності;

• рейтинг кінцевого виконавця заявки – використовується як ваговий коефіцієнт для раціонального розподілу ресурсів в підсистемах.

Кожна з заявок у вхідному потоці даних отримує динамічний пріоритет, залежно від представлених набору ознак і поточного стану всієї СМО в цілому. Приймемо, що всі обчислювальні вузли К, в рамках обраного класу завдання, ідентичні і будь-яка заявка може бути обслужена будь-яким з них. При цьому на кожному обчислювальному вузлі для ефективного обслуговування заявок застосовуються відносні пріоритети.

Враховуючи особливості кожного сервісу, формалізуємо характеристики побудованої моделі. Кількість джерел I, і інтенсивність μ_n=1, ..., I безпосередньо залежить від кількості користувачів, що звертаються в даний момент до хмари, при чому у разі одночасного звернення одного клієнта до різних рівнів підсистем будемо вважати як заявки, що надійшли від двох незалежних один від одного джерел. Враховуючи це, інтенсивність надходження заявок в хмарну систему в цілому буде нерівномірною не залежно від обраного інтервалу часу моделювання. Крім цього, в СМО хмари можна виділити кілька фаз F обслуговування заявок. Це обумовлено архітектурою технічного рішення, що дозволяє масштабувати потужності залежно від поставлених завдань. Хмарний контролер, що керує розміщенням обчислювальних задач на запущених додатках, а так само запуском і зупинкою обчислювальних вузлів, здатний визначати класи завдань, що дає можливість використовувати гнучке управління потоками запитів.

Визначимо схему управління потоками запитів (рис. 1) і виділимо три фази обслуговування заявок: накопичення заявок в контролері хмарної системи (фаза 1), приорітетне обслуговування заявок на обраному обчислювальному вузлі (фаза 2), генерація пакетів даних, запитаних користувачами (фаза 3).

В рамках дослідження встановлено, що для потоку запитів до хмарної інформаційної системи характерна сильна нерівномірність інтенсивності надходження заявок і пакетів. Заявки не рівномірно розподілені на часовому інтервалі, і групуються в одних інтервалах, і повністю відсутні в інших.

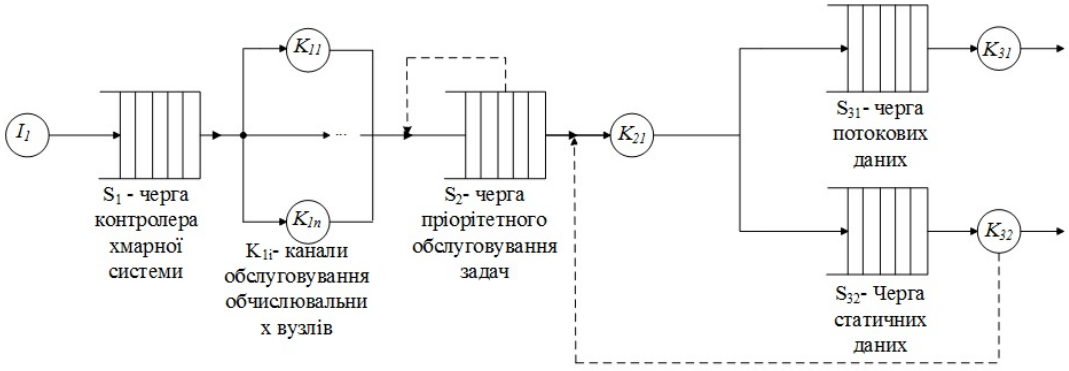

Рис. 1 Схема управління потоками запитів в системі з хмарними обчисленнями

При цьому для кожного класу задач випадковий процес надходження в систему запитів характеризується законом розподілу, що встановлює зв'язок між значенням випадкової величини і ймовірністю появи цього значення. Такий потік може бути описаний ймовірнісною функцією розподілу інтервалів часу між сусідніми запитами. На підставі статистичних даних про кількість і розміри запитів, що надходять а так само характеристики інтервалів часу між запитами отримано наступні відповідності:

– для додатків, що відносяться до першого класу і здійснюють обробку потокових даних (передача відео трафіку) характерний розподіл Парето;

– для додатків, що відносяться до другого класу і здійснюють обробку статичних даних (передача бінарного трафіку) характерно розподіл Вейбулла між надходять на обслуговування заявками;

– для додатків, що відносяться до третього класу і здійснюють обробку статичних даних (передача трафіку даних Найбільшого розміру) характерний Хі- квадрат-розподіл.

Особливістю архітектури віртуалізації є її масштабованість і реконфігурованість. Тому основним завданням управління є вибір необхідної кількості обчислювальних ресурсів в кожен наступний момент часу роботи хмарної системи. При організації доступу до мультимедійних ресурсів це особливо актуально, так як створюване навантаження на сервіси може змінюватися в досить короткі інтервали часу [14, 15].

Для того, щоб запобігти вичерпання ресурсів у вже запущених додатках і підготувати додаткові обчислювальні потужності потрібно, динамічно складати план стану ресурсів і застосовувати його для оптимізації структури хмари.

Одним з процесів створення віртуальної машини є планування. У результаті обробки в системі задається передбачувана кількість користувачів, а також може бути створений виділений шаблон віртуальної машини з необхідними апаратними характеристиками і програмним забезпеченням для проведення міграції. На основі отриманого шаблону і даних про попередні аналогічні заходи сервером розкладів здійснюється розрахунок конфігурації для розгортання сервісу. При цьому у разі ідентичності віртуальних машин з програмного забезпечення пропонуються варіанти вже створених раніше образів, що зберігаються в NAS. Для зручності конфігурації віртуальної машини пропонується три види.

Перша конфігурація – забезпечить запас продуктивності у разі непередбачуваного збільшення кількості користувачів. Коефіцієнт масштабування при цьому розраховується динамічно. При цьому хмарна система виконує масштабування в рамках всіх доступних на поточний момент часу ресурсів. Крім того, системою надається можливість вибрати діапазон кількості примірників віртуальних машин, доступних для запуску.

Друга конфігурація – забезпечить меншу продуктивність віртуальної машини, у порівнянні з заданою кількістю користувачів. Однак такий шаблон є найбільш ефективним при використанні сервісів для вузькоспеціалізованої аудиторії користувачів. Він дозволяє скоротити накладні витрати при порівняно малій кількості користувачів, щодо заявленого числа передплатників. Як і у випадку з першим варіантом шаблону, можливо задати кількість примірників віртуальних машин, що дозволяє забезпечити паралельний запуск на декількох обчислювальних вузлах в умовах нестачі ресурсів у хмарній системі.

Третій варіант конфігурації – створюється з використання заданих користувачем характеристик, що включають в себе фіксований коефіцієнт масштабування, і фіксовану кількість примірників віртуальних машин, які будуть запущені незалежно від кількості користувачів відразу після початку процесу міграції.

Іншим процесом є обслуговування запитів користувачів і масштабування ресурсів в рамках роботи додатків. Система враховує загальну кількість запитів від кожного з джерел, що дає можливість прогнозування навантаження не тільки на хмару і запущені програми, але і на окремі сегменти ЛВС. На основі отриманих даних системою управління хмарою відповідно до заданого планом здійснюється міграція і масштабування обчислювальних ресурсів.

Іншим процесом є обслуговування запитів користувачів і масштабування ресурсів в рамках роботи додатків. Система враховує загальну кількість запитів від кожного з джерел, що дає можливість прогнозування навантаження не тільки на хмару і запущені програми, але і на окремі сегменти ЛВС. На основі отриманих даних системою управління хмарою відповідно до заданого планом здійснюється міграція і масштабування обчислювальних ресурсів.

Таким чином, запропоновані методи доступу і керування хмарною системою дозволяють підвищити ефективність її використання в умовах обмеженості ресурсів, а також звести до максимуму кількість користувачів.

5. Результати досліджень

У ході експерименту проводився аналіз спільної роботи всіх запропонованих алгоритмів при обробці потоків запитів різної інтенсивності до всіх доступних додатків інформаційної системи. При цьому на кожному етапі експерименту встановлювалося обмеження на кількість виділених ресурсів не тільки на кожну з підсистем СДО, а і на обчислювальні вузли в цілому. Це дозволило визначити оптимальну конфігурацію апаратного обладнання, необхідного для підтримки всієї системи.

Експериментальна апробація проводилася на реальній хмарної системі OpenStack [16] з використанням додаткового модуля розподілу навантаження реалізованого на мові Python.

Для наближення результатів до реальних даних при проведенні представленних експериментів створювався потік запитів, аналогічний знятому трафіку інформаційної системи для наукових досліджень. При цьому кількість одночасних запитів, що надійшла в систему, склала 10000, що відповідає кількості активних користувачів в ІСНД (інформаційній системі наукових досліджень), згідно з проведеним дослідженням. Всі запити класифіковані на 6 груп, що характеризують типову поведінку користувачів. Для кожної з групи користувачів по рівневої моделі ресурсів системи, згідно отриманим законам розподілу задана інтенсивність використання кожного з компонентів системи, а також обсяг затребуваних даних. Експеримент проводився на інтервалі часу рівному одній годині, що відповідає найбільш тривалому періоду часу пікового навантаження системи, зафіксованому в реальному трафіку. У табл. 1 представлені основні показники, отримані в результаті експерименту для першої групи користувачів.

У ході проведеного експерименту встановлено, що використання розроблених методів управління віртуальними ресурсами, дозволяє знизити кількість відмов в обслуговуванні на 11–15 % при зверненні користувачів до системи. Додатково проведена оцінка методів управління щодо балансування навантаження між віртуальними обчислювальними вузлами. Оцінка ефективності балансування навантаження між віртуальними ресурсами проводилася за нааступними критеріями: час відгуку сервера і кількість виділених віртуальних серверів для кожної з підсистем. На рис. 2 представлений графік роботи системи балансування навантаження на обчислювальних ресурсах ІСНД.

Таблиця 1

Ефективність обслуговування запитів.

Підсистеми ІСНД, тип даних	текст	файли	відео- аудіотрафік
Загальна кількість запитів	8000	1000	1000
Об'єм даних	32658	9330	10340
Кількість запитів, що обслуговувались	5443 (4352)	622 (418)	517 (356)
Інтенсивність обслуговування, запитів/с	90,71 (72,53)	10,36 (6,96)	8,61 (5,93)

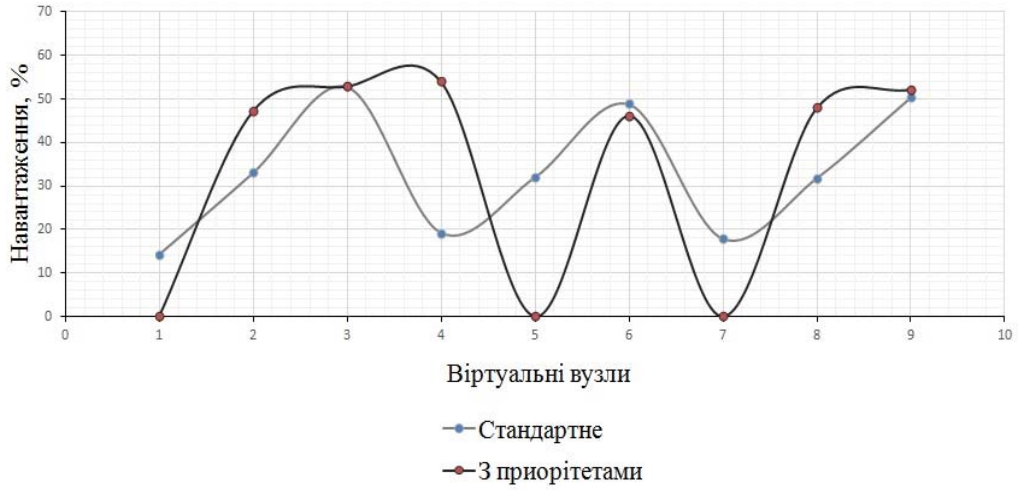

Рис. 2. Балансування навантаження мід віртуальними вузлами

Проведене дослідження показало, що при використанні розроблених методів оптимізації для виділення ресурсів для всіх запущених екземплярів додатків, що входять до ІСНД, можливо отримання до 20 % вільних ресурсів (віртуальних серверів) і гарантовано забезпечення спільної роботи, що задовольняє вимогам потенційних користувачів.

6. Висновки

Запропоновані в дослідження методи керування хмарною системою дозволяють підвищити якість і ефективність її використання при обмежених ресурсах, а також збільшими кількість користувачів до максимуму.

Література

1. Armbrust, M. Above the Clouds: A Berkeley view of cloud computing [Text] / M. Armbrust, A. Fox, R. Griffith // Science. – 2009. – P. 191–196.

2. Мацуєва, К. А. Методи управління ресурсами і додатками в обчислювальних системах на базі хмарних обчислень [Текст]: тез. доп. / К. А. Мацуєва // Десята міжнародна науково-практична конференція "Математичне та імітаційне моделювання систем. МОДС '2015". – Чернігів, 2015. – С. 89.

3. Buyya, R. Cloud Computing. Principles and Paradigms [Text] / R. Buyya, J. Broberg, A, Goscinski. – John Wiley, 2011. – 675 p. doi: 10.1002/9780470940105

4. Blazewicz, J. Eugene Levner Handbook on Scheduling. From Theory to Applications [Text] / J. Blazewicz. – Berlin : Springer, 2007. – 647 p.

5. Jones, W. M. Beowulf Mini-grid Scheduling [Electronic Resource]/ W. M. Jones, L. W. Pang. – CiteSeerX. – The Pennsylvania State University. – Available at: http://citeseer.ist.psu.edu/696342.html

6. Aida, K. Job Scheduling Scheme for Pure Space Sharing among Rigid Jobs [Text] / K. Aida, H. Kasahara, S. Narita // Lecture Notes In Computer Science, Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing. – London: Springer-Verlag, 1998. – P. 98–121. doi: 10.1007/bfb0053983

7. Sinnen, O. Communication contention in task scheduling [Text] / O. Sinnen, L. A. Sousa // IEEE Transactions on Parallel and Distributed Systems. – 2005. – Vol. 16, Issue 6. – P. 503–515. doi: 10.1109/tpds.2005.64

8. Гергель, В. П. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора [Текст] / В. П. Гергель, П. Н. Полежаев // Вестник Нижегородского университета им. Н. И. Лобачевского. – 2010. – № 5 (1). – С. 201–208.

9. Feitelson, D. Job Scheduling Strategies for Parallel Processing [Text] / Feitelson D., Rudolph L. Metrics and Benchmarking for Parallel Job Scheduling. – Orlando: Springer, 1998. – P. 1–24.

10. Brucker, P. Scheduling Algorithms [Text] / P. Brucker. – Berlin: Springer, 2007. – 371 p. doi: 10.1007/978-3-662-03088-2

11. Bender, M. A. Communication-Aware Processor Allocation for Supercomputers [Text] / M. A. Bender, D. P. Bunde, E. D. Demaine, S. P. Fekete, V. J. Leung, H. Meijer, C. A. Phillips // Lecture Notes in Computer Science. – 2005. – Vol. 3608. – P. 169–181. doi: 10.1007/11534273_16

12. Pinedo, M. L. Planning and Scheduling in Manufacturing and Services [Text] / M. L. Pinedo. – LLC: Springer Science+Business Media, 2009. – 506 p. doi: 10.1007/978-1-4419-0910-7_14

13. Петров, Д. Л. Оптимальный алгоритм миграции данных в масштабируемых облачных хранилищах [Текст] / Д. Л. Петров // Управление большими системами. – 2010. – № 30. – С. 180–197.

14. Borodin, A., El-Yaniv R. Online computation and competitive analysis [Text] / A. Borodin, R. El-Yaniv // Cambridge University Press, NewYork. – 1998. – Vol. 53.

15. Aspnes, J. Competitive analysis of distributed algorithms [Text] / J. Aspnes // Lecture Notes in Computer Science. – 1998. – P. 118–146. doi: 10.1007/BFb0029567

16. OpenStack Open Source Cloud Computing Software [Electronic Resource]. – Available at: http://www.openstack.org/

References

1. Armbrust, M., Fox, A., Griffith, R. (2009). Above the Clouds: A Berkeley view of cloud computing, Science, 1, 191–196.

2. Matsueva, K. (2015). Methods of resource management and applications in computing systems based on cloud computing. Tenth International Scientific Conference Mathematical and simulation systems. MODS' 2015, 89–90.

3. Buyya, R. (2011). Cloud Computing. Principles and Paradigms. John Wiley, 675. doi: 10.1002/9780470940105

4. Blazewicz, J. (2007). Handbook on Scheduling. From Theory to Applications. Berlin: Springer, 647.

5. Jones, W. M. (2014). Beowulf Mini-grid Scheduling. The Pennsylvania State University. Available at: http://citeseer.ist.psu.edu/696342.html

6. Aida, K., Kasahara, H., Narita, S. (1998). Job scheduling scheme for pure space sharing among rigid jobs. Lecture Notes in Computer Science, 98–121. doi: 10.1007/bfb0053983

7. Sinnen, O., Sousa, L. A. (2005). Communication contention in task scheduling. IEEE Transactions on Parallel and Distributed Systems, 16 (6), 503–515. doi: 10.1109/tpds.2005.64

8. Sinnen, O. (2005). Communication contention in task scheduling. IEEE Transactions on Parallel and Distributed Systems, 16 (6), 503–515.

9. Gergel, V. P. (2010). Research of scheduling algorithms of parallel tasks for cluster computing systems using the simulator. Bulletin of the Nizhny Novgorod University N. I. Lobachevskogo, 5 (1), 201–208.

10. Brucker, P. (2007). Scheduling Algorithms.Berlin. Springer, 371. doi: 10.1007/978-3-662-03088-2

11. Bender, M. A., Bunde, D. P., Demaine, E. D., Fekete, S. P., Leung, V. J., Meijer, H., Phillips, C. A. (2005). Communication-Aware Processor Allocation for Supercomputers. Lecture Notes in Computer Science, 3608, 169–181. doi: 10.1007/11534273 16

12. Pinedo, M. L. (2009). Planning and Scheduling in Manufacturing and Services. LLC: Springer Science. Business Media, 509. doi: 10.1007/978-1-4419-0910-7_14

13. Petrov, D. (2010). Optimal algorithm of data migration in scalable cloud storages. System management, 30, 180–197.

14. Borodin, A., El-Yaniv, R. (1998). Online computation and competitive analysis. Cambridge University Press, NewYork, 53.

15. Aspnes, J. (1998). Competitive analysis of distributed algorithms. Lecture Notes in Computer Science, 118–146. doi: 10.1007/bfb0029567

16. OpenStack Open Source Cloud Computing Software. Available at: http://www.openstack.org/

Рекомендовано до публікації д-р техн. наук Литвинов В. В.
Дата надходження рукопису 24.06.2015

Мацуєва Карина Андріївна, аспірант, асистент, кафедра комп'ютеризованих систем управління, Національний авіаційний університет, пр. Космонавта Комарова, 1, м. Київ, Україна, 03058
E-mail: karyna_matsueva@bigmir.net

УДК 534.8; 534.321.9; 621.8.034
DOI: 10.15587/2313-8416.2015.46987

ИНТЕНСИФИКАЦИЯ ТЕПЛОМАССООБМЕННЫХ ПРОЦЕССОВ В АППАРАТАХ ПОГРУЖНОГО ГОРЕНИЯ КОЛЕБАНИЯМИ КОНТАКТИРУЮЩИХ ФАЗ

© В. Е. Никольский

Исследован вопрос создания колебаний в контактирующих фазах (газ – жидкость) как средства интенсификации тепломассообменных процессов при барботажном режиме в аппаратах погружного горения (АПГ). Определены кинематические и силовые характеристики в волновом поле. Предложено размещать резонирующие барботажные устройства в зоне выхода продуктов сгорания из АПГ и входа их в жидкость. Исследования на лабораторных и действующих технологических установках подтвердили перспективность выбранного направления
Ключевые слова: *аппарат погружного горения, тепломассообменные процессы, колебания, барботажный режим*

The problem of oscillation generation in contacted gas – liquid phases as the means of intensification of heat-mass exchange processes at bubbling of the gas – liquid system in the immersed burning apparatus (IBA) was studied. Kinematic and force characteristics in the wave field were determined. It was proposed to place the resonant bubbling devices in the outlet zone of burning products from IBA and at their entry to liquid. The experiments on laboratory-scale plants and technological plants show great potential of proposed apparatus
Keywords: *immersed burning apparatus, heat-mass exchange processes, oscillation, bubbling*

1. Введение

В статье рассмотрен аспект вопроса создания колебаний в контактирующих фазах газ – жидкость как средства интенсификации тепломассообменных процессов при прямоточном барботажном режиме взаимодействия фаз в газо-жидкостной системе в аппаратах погружного горения.

Аналитически определены кинематические и силовые характеристики в волновом поле, порождаемым взаимодействием фаз.

2. Постановка проблемы

Скорость технологических процессов, в том числе реакционно-разделительных, протекающих в диффузионной области, определяется интенсивностью переноса вещества и энергии. Одним из путей интенсификации этих процессов является турбулизация контактирующих фаз на поверхности их раздела. Известно, что в турбулентном потоке элементарные струи изменяют скорость и направление, в связи с чем в каждой точке объема происходит пульсация скорости. Наложение колебаний извне в некоторых областях частот, соответствующих собственной частоте колебаний элементов структуры, приводит к интенсификации процесса переноса. Однако, литературные данные по этому вопросу весьма противоречивы как в оценке причин этого эффекта, так и в части оценки результатов интенсификации.