УДК 681.324

С.В. Мінухін, О.В. Мєзєнцев

Харківський національний економічний університет, Харків

ДОСЛІДЖЕННЯ ПЛАНУВАЛЬНИКА РЕСУРСІВ MAUI ТА МОЖЛИВОСТЕЙ ІНТЕГРАЦІЇ ДО ЙОГО СКЛАДУ НОВИХ АЛГОРИТМІВ

Проаналізовано локальні системи управління розподіленими обчисленнями та методи планування ресурсів в кластерних системах. Проаналізовано алгоритми планування ресурсів на прикладі планувальника Maui. Досліджено вихідний код планувальника ресурсів. Наведені та проаналізовані особливості програмної реалізації нового алгоритму на основі методу покриття у складі планувальника Maui. Досліджено поширення можливостей планувальника Maui шляхом інтеграції в програмне забезпечення нових планувальників для підвищення ефективності його роботи.

Ключові слова: *ресурси, кластер, система управління пакетного оброблення завдань, PBS, Torque, Maui, планувальник, найменше покриття.*

Вступ

При проектуванні розподіленої обчислювальної системи (кластеру) постає питання вибору оптимального алгоритму планування черги обчислювальних завдань, який ефективно розподіляє ресурси між завданнями, які запускаються на цій системі.

Зазвичай планувальник обирається розробником чи адміністратором кластера, виходячи з власного досвіду, досвіду користувачів або шляхом натурного випробування, необхідно проводити його тривалий період часу. На кластері, який використовується реальними користувачами, відсутня можливість порівняти роботу планувальників на одному наборі завдань, а тому й складно застосувати формальний критерій для порівняння, який залежить від конкретного набору завдань. Крім того, результати порівняння роботи планувальників, отримані в ході натурних випробувань на одному кластері, можуть бути не застосовні на іншому.

Якість надання сервісу з високопродуктивних обчислень можна визначити різними шляхами. Це може бути наявність різноманітного прикладного програмного забезпечення, зручність керування власною чергою завдань та файлами, швидкість обміну даними між кластером та терміналом користувача, надійність роботи кластеру в цілому тощо, але ефективність функціонування кластеру визначається, в першу чергу, узгодженістю розподілу наявних ресурсів, яке має відбуватися з використанням процесів планування обчислювальних процесів.

Кластер з необмеженою кількістю вузлів повинен миттєво обробляти запит користувача на отримання ресурсів і одразу запускати завдання на виконання. Зрозуміло, що в реальній ситуації так не відбувається. Реальний кластер має систему черг задач та спеціальні алгоритми для керування чергами. Кожен алгоритм по-своєму визначає приорітетність запуску кожного завдання [1].

Метою даної роботи є дослідження можливостей інтеграції нових планувальників в програмне забезпечення планувальника Maui для підвищення ефективності процесів планування та функціонування всієї розподіленої системи цілому.

Програмна база кластера

В якості базового програмного забезпечення модельованої системи в даній роботі обрані такі програмні продукти, як менеджер ресурсів Torque, локальний планувальник Maui та операційна система сімейства Unix - CentOS.

Torque – це менеджер ресурсів, який є однією з версій Portable Batch System, який відповідає за відстеження доступної кількості ресурсів на вузлах кластера та запуск завдань. Torque управляє завантаженням обчислювальних комплексів, що складаються з певної кількості обчислювальних вузлів, що працюють під управлінням операційної системи сімейства Unix. Torque має вбудований планувальник завдань, він являє собою службу у ОС сімейства Unix (далі демон) яка має назву pbs_sched, що визначає момент запуску завдань. Менеджер ресурсів забезпечує низькорівневі функціональні можливості такі, як запуск, утримування (тимчасове припинення), скасування і контроль виконання завдання. Без цих можливостей менеджера ресурсів планувальник не зможе самостійно контролювати виконання завдання.

Torque кластер складається з одного головного вузла та багатьох обчислювальних вузлів. На головному вузлі працює демон pbs_server (сервер Torque), а на обчислювальних вузлах запускається демон pbs_mom. Клієнтські команди для подання та управління завданнями можуть бути встановлені на будь-якому комп'ютері (в тому числі й на хостах, де відсутні демони pbs_server або pbs_mom) [2, 10].

На головному вузлі також працює і демон планувальника. Планувальник взаємодіє з pbs_server для прийняття рішень, пов'язаних з локальними політиками щодо використання ресурсів і виділення вузлів для виконання завдань. Простий планувальник завдань FIFO, а також код для побудови більш складних планувальників наданий у дистрибутиві вихідного коду (source code). У

більшості випадків користувачі Torque використовують планувальники з біл ьш широким функціоналом, такі, як Maui або Moab. У цій роботі розглядається планувальник Maui.

Користувачі відправляють завдання на сервер Torque (pbs_server) за допомогою команди qsub. Коли pbs_server отримує нове завдання, він інформує про це планувальник. Коли планувальник знаходить вільні вузли для виконання певного завдання, він посилає інструкції для виконання завдання на вузлі зі списку вузлів pbs_server. Потім pbs_server посилає нове завдання на перший вузол у списку вузлів і дає йому інструкції зі запуску завдання. Цей вузол визначається як виконавчий хост і зветься Mother Superior. Інші вузли в ході виконання завдання називаються Sister Moms [2].

Maui – планувальник завдань, з розширеним функціоналом у паралельних обчислювальних системах. Він опитує Torque на предмет наявності вільних ресурсів і завдань у черзі, які необхідно виконати. На основі отриманих даних і своїх налаштувань він приймає рішення про запуск якого-небудь завдання і посилає команду серверу Torque виконати її. Maui дозволяє гнучко налаштовувати різні стратегії заповнення кластера, пріоритети для завдань за різними критеріями: пріоритетів, кількістю запитуваних ресурсів, тощо [3].

Як і для всіх інших планувальників, що застосовуються в системах пакетного оброблення даних (далі – СПО), функції Maui наступні: при заданій множині готових до виконання завдань (з черги) і для поточного стану ресурсів визначається черговість запуску завдань та знаходяться відповідні ресурси для тих з них, які можна запустити в даний момент. У цих рамках Maui реалізує нові механізми, засновані на "передбаченні майбутнього" і спрямовані на оптимізацію управління виконанням. Крім того, в Maui є ряд нововведень, що відрізняють його від інших планувальників:

1) алгоритми зворотного заповнення (backfill) і справедливого розподілу ресурсів (fairshare) для підвищення ефективності системи та зменшення часу очікування в черзі;

2) система автоматичного визначення пріоритетів завдань, що дозволяє давати ключовим користувачам перевагу при розподілі ресурсів;

3) покращена діагностика проблем з завданнями, вузлами і програмними компонентами СПО;

4) розширений спектр статистики. За допомогою Maui адміністратор може отримувати повну історичну (хронологічну) та поточну інформацію про завдання, чергу, планувальника, стан системи тощо;

5) в Maui є можливість моделювання роботи СПО, за допомогою якої можна перевірити налаштування планувальника "у дії". Це дозволяє підібрати конфігурацію системи, яка найбільш повно відповідає вимогам конкретної виробничої обстановки [4].

У Maui є ще одна істотна відмінність: звичайні планувальники не мають власних командних інтерфейсів внаслідок того, що вони їм не потрібні. В Maui повністю реалізовані функції резервування, він має в своєму розпорядженні і відповідний інтерфейс для зовнішнього або, так званого, адміністративного резервування.

Аналіз процесів планування ресурсів на кластері під управлінням планувальника Maui

Як і багато інших планувальники, Maui працює ітераційно, тобто перемежовуючи процес планування з очікуванням або виконанням зовнішніх команд. Кожен цикл починається при здійсненні однієї з таких подій:

1) змінюється стан завдання або ресурсу;
2) досягнуто межу резервування;
3) отримана зовнішня команда;
4) з початку попереднього циклу минув час, визначений як максимальний.

Поняття резервування формалізується в Maui відповідним типом об'єктів.

Об'єкт "резервування" складається з:

1) ACL (Access Control List, список контролю доступу) – набору параметрів, керуючись яким планувальник визначає, для яких завдань доступні зарезервовані ресурси;
2) списку зарезервованих ресурсів;
3) часу дії резервування.

На рівні реалізації кожне резервування фізично являє собою запис у базі даних Maui. Об'єкт резервування прив'язаний, з одного боку, до часу, причому до майбутнього. З іншого боку, інтерпретація резервувань передбачає розгляд станів ресурсів на моменти початку резервувань. Зображується це у вигляді тимчасової розгортки (time line) станів ресурсів: для кожного ресурсу на вісі часу ставляться мітки, що визначають, яким завданням в цей момент ресурси можуть бути доступні. Сукупність певної кількості міток і утворюють одне резервування (тому зарезервувати можна цілий комплекс ресурсів, і всі вони будуть належати одному резервування) [4].

Зауважимо, що попереднє резервування проводиться під завдання, якого ще немає в СПО. Отже повинен бути механізм, який дозволяє зв'язати завдання зі зробленим резервуванням. Об'єкт резервування має список контролю доступу ACL, і саме відповідно з ним відбувається прив'язка завдання до певного резервування. Зарезервовані ресурси можуть отримати тільки такі завдання, які мають хоча б один з параметрів: GROUPLIST, USERLIST, ACCOUNTLIST, CLASSLIST і QOSLIST, що збігається за значенням з відповідними атрибутами ACL резервування. Тобто, прив'язка може здійснюватися за реєстраційним іменем користувача, його групи, класу та якості обслуговування. Можливості налаштування списків доступу ACL в Maui досить великі і, зокрема, можна домогтися того, щоб зарезервовані ресурси були доступні тільки для певного завдання, причому

навіть ще не надійшовши у чергу СПО (саме це потрібно для метадиспетчеризації) [9].

Maui дозволяє зарезервувати 4 найбільш важливих типи ресурсів:

1) [PROCS = <INTEGER>:];
2) [MEM = <INTEGER>:];
3) [DISK = <INTEGER>:];
4) [SWAP = <INTEGER>:].

Кількість ресурсів визначається певним завданням. У завдання може входити кілька паралельних задач, кожне з яких виконується на одному вузлі. Не слід плутати завдання та задачі (в документації Maui використовуються, відповідно, job і task). Кожне завдання може складатися з кількох задач. Задача може виконуватися тільки на одному вузлі та є, як би, квантом завдання. Коли здійснюється резервування під завдання, то в запиті вказуються ресурси, необхідні для одного завдання, і кількість задач. Таким чином, всі завдання видаються однотипними, принаймні, які вимагають однакову кількість ресурсів. Далі розглядатимемо ті завдання, які мають тільки одну задачу.

В Maui передбачені засоби для операцій з існуючими резервуваннями. За допомогою команди showres можна отримати інформацію про те, де, для яких завдань, на який час і скільки ресурсів зарезервовано, причому доступний пошук, як по резервуванню, так і по вузлах. Існує також команда releaseres для скасування створених резервувань [4].

В процесі планування застосовується ще один, внутрішній тип резервування Maui – job-резервування. З точки зору реалізації він мало відрізняється від описаного вище адміністративного резервування.

Воно використовується в двох випадках:

- для забезпечення доступності ресурсів під виконуються завдання. Як тільки завдання запускається, то відразу, на весь час виконання (walltime), створюється job-резервування, що закриває доступ до ресурсів, необхідним для цього завдання, – всім, крім нього самого. Може статися, що під час роботи всі ці ресурси і не будуть використовуватися, однак як тільки вони будуть потрібні завданням, вони гарантовано зможуть їх отримати. Коли завдання завершується (можливо, раніше, ніж через час walltime) job-резервування скасовується, і ресурси стають вільними [4];

- job-резервування застосовується для того, щоб завдання з більш високими пріоритетами не були затримані запуском менш пріоритетних завдань, що може трапитися при роботі алгоритму backfill. В загальному вигляді процес запуску завдань виглядає так: до тих пір, поки це можливо, впорядковані за пріоритетами завдання беруться з черги і запускаються (попутно створюється job-резервування); як тільки чергове завдання не можна запустити через брак (відсутність) ресурсів для нього, визначається найближчим часом, в яке можна буде провести запуск, і починаючи з нього створюється job-резервування затребуваних ресурсів на час walltime.

Після того як зроблена деяка, сконфігурована кількість резервувань, починає роботу власне алгоритм backfill. Він з'ясовує, які вузли і на який час, починаючи з поточного, вільні (це як раз визначається зробленим job-резервуванням). Після цього вільні вузли об'єднуються у «вікна». Сумарна «ширина» «вікон» може виявитися більше кількості вільних в даний момент вузлів, тому що деякі вузли можуть входити в кілька вікон. Потім з усіх вікон вибирається одне, як правило, саме широке. Серед усієї решти завдань вибирається і запускається те завдання, яке найбільш точно задовольняє цьому «вікну». Якщо є можливість, то запускається не одне завдання. Так як при запуску завдань алгоритмом backfill враховуються зроблені job-резервування, то відповідні їм завдання не будуть затримані [4].

Основною відмінністю job-резервування від адміністративного є те, що воно здійснюється самим планувальником і ним же відміняється перед початком чергового кроку планування. Причому картина резервувань може відрізнятися від попереднього кроку планування досить істотно. Це має місце у випадку, коли надходять нові завдання, що міняють порядок в черзі або звільняють ресурси. На відміну від адміністративних, job-резервування не гарантує запуск завдання в зазначений час. Таким чином, job-резервування (не для запущених завдань) є всього лише мітками, що дозволяють коректно врахувати в алгоритмі backfill пріоритети завдань. Адміністративні та job-резервування існують паралельно, тобто job-резервування можуть накладатися на адміністративні, якщо звичайне завдання, під яке робиться job-резервування, задовольняє ACL адміністративного [5].

Розглянемо планування на основі пріоритетів, яке передбачає, що ресурси для більш пріоритетних завдань виділяються раніше, ніж для менш пріоритетних. В цих умовах широко використовуються алгоритми типу FCFS (FIFO), які працюють за принципом виділення вільних ресурсів самому пріоритетному завданню з черги, яке може на них розміститися. Цей тип планування використовує базовий планувальник Torque (демон pbs_sched). При цьому виходить, що більша частина процесорів буде завжди зайнята дрібними завданнями, тобто виникає фрагментація ресурсів. Навіть якщо завдання має найвищий пріоритет, необхідний йому обсяг ресурсів може ніколи не утворитися і, отже, завдання може ніколи не стартувати. Для середовища, яке обслуговує однопроцесорні завдання, цієї проблеми немає. Вона виникає, коли є колективні ресурси, наприклад, при обслуговуванні багатопроцесорних завдань. Аналогічна ситуація виникає на ресурсах із загальною пам'яттю, в середовищі з загальним файловим простором та інших випадках, коли ресурси діляться між завданнями, а не виділяються під завдання цілком.

Для вирішення проблеми коалокації в локальних системах, наприклад, в планувальнику Maui, використовується алгоритм зворотного

заповнення backfill [6], розроблений для великих багатопроцесорних систем (MPP) типу IBM SP2.

Він має такі переваги:

1) в умовах роботи в пріоритетною системі дозволяє уникнути зависання завдання, гарантуючи його запуск;

2) ефективно завантажує ресурси, дозволяючи уникнути їх фрагментації;

3) має прийнятні часові характеристики при роботі на великій кількості обчислювальних вузлів;

4) дозволяє працювати на множині гетерогенних ресурсів.

Алгоритм зворотного заповнення backfill працює за наступним принципом: розміщуючи найбільш пріоритетне завдання, він визначає момент часу, коли звільниться достатня кількість ресурсів, зайнятих завданнями, які вже виконуються, і виконує резервування цих ресурсів. Завдання з меншим пріоритетом може бути запущено поза чергою, але тільки в тому випадку, якщо воно не буде заважати запуску всіх (в консервативному варіанті backfill) більш пріоритетних завдань. Відзначимо, що в опублікованих роботах за алгоритмом backfill міститься лише згаданий вище принцип, а також загальна схема процесу розподілу завдань з черги по ресурсах.

Однак варто зауважити, що алгоритм backfill не є ідеальним і також має свої недоліки. Головним з них є те, що коли всі завдання вже розподілені на ресурси кластеру і виконуються, нові завдання, які надходять до черги у кластер, не розподіляються за рівномірним законом(на перший ресурс, якій звільнився) на ресурси кластера, а потрапляють на перший ресурс зі всього списку ресурсів, тобто у його початок. Цей недолік може зменшити продуктивність та швидкодію виконання завдань. Можливим рішенням цієї проблеми може бути доопрацювання планувальника Maui шляхом впровадження в нього додаткового алгоритму планування, який би забезпечив усунення цього недоліку [7].

Аналіз вихідного коду планувальника Maui

Вихідний код планувальника є у вільному доступі (www.adaptivecomputing.com). Проаналізувавши дистрибутив планувальника Maui, можна зробити висновок, що він написаний мовою програмування C. Також, він заснований на базі планувальника Moab, який зараз є комерційним продуктом. Також було виявлено той факт, що основним алгоритмом планування у Maui (Moab) є backfill. Можна навіть сказати, що Maui (Moab) і є його реалізацією з допоміжними механізмами планування такими, як політики справедливого розподілу ресурсів, Розробка планувальника Maui була завершена у 2003 р., коли Moab перевели у стадію комерційного продукту. Maui має всі можливості планувальника Moab станом на 2003 р., тобто за всі функції управління процесом планування відповідає вихідний код Moab. Зараз Moab має більш потужний функціонал по управлінню процесами планування на кластері.

Також, при аналізі вихідного коду планувальника Maui були помічені спроби його модифікації іншими організаціями шляхом підключення сторонніх алгоритмів виділення ресурсів, та політик справедливого розподілу ресурсів. Отже можна зробити висновок, що можлива інтеграція нового алгоритму планування ресурсів [7].

Сутність нового алгоритму планування та особливості його інтеграції до складу Maui

Основним алгоритмом планування у Maui є backFill (алгоритм зворотного заповнення), але як і кожний інший алгоритм, він має свої недоліки. Враховуючи те, що кластерні системи зазвичай мають достатньо високий півень гетерогенності, було прийнято рішення інтегрувати до локального планувальника ресурсів Maui новий алгоритм планування, який буде ефективним при умовах високого рівня гетерогенності кластеру, та високої інтенсивності надходження завдань у систему.

Основна ідея, покладена в основу нового алгоритму, полягає у використанні в якості процедури планування вирішення завдання про найменше покриття (ЗНП).

Суть методу ЗНП полягає у тому, щоб спланувати вирішення якомога більше завдань, залучуючи при цьому якомога менше ресурсів.

Вхідними даними для вирішення ЗНП є матриця відповідності. Ця матриця відображає дані про те, яке завдання на якому ресурсі може бути вирішено з урахуванням, що на одному ресурсі може виконуватися декілька завдань (рис. 1) [7].

Завдання	Ресурси							
	R_{1*}	R_2	R_{3*}	R_4	R_{5*}	R_6	R_7	R_8
$task_1$	1							
$task_2$		1	1	1				
$task_3$	1					1		
$task_4$			1					1
$task_5$		1	1	1				
$task_6$	1						1	
$task_7$					1			1
$task_8$					1		1	
$task_9$					1			
$task_{10}$		1	1					1
$task_{11}$		1			1			
$task_{12}$	1			1		1		

Рис. 1. Матриця відповідності ресурси-завдання

Для побудови цієї матриці пропонується програмно використовуватися функції з вихідного

коду планувальника Maui. Планувальник є дуже складною системою з безліччю параметрів та факторів, які враховуються при плануванні, також у нього дуже тісно інтегрований алгоритм планування backfill, що дуже ускладнює інтеграцію нового алгоритму. Тому, щоб не вносити евристику у роботу планувальника, пропонується виконувати процедуру так званого «матчінгу» – визначення того, на якому ресурсів може виконуватися те чи інше завдання – функцією самого планувальника, а в якості параметрів передавати їй завдання, наявні ресурси, та вимоги до завдань.

Після формування цієї матриці вирішується ЗНП. В процесі реалізації алгоритму планування також буде реалізовано алгоритм оброблення виключень, тобто обробка тих завдань, які по тій чи іншій причині не пройшли процедуру планування. Результатом вирішення ЗНП є вектор призначень, тобто, яке завдання на якому ресурсі буде вирішуватися.

Відладка планувальника Maui відбувається у інтегрованому середовищі розробки Netbeans та Eclipse. Факт того, що планувальник Maui дуже тісно інтегрований з алгоритмом планування backfill істотно ускладнює процес інтеграції нового алгоритму. Зміни в одному місті вихідного коду ведуть до значних проблем та помилок у всьому коді.

Без повного розуміння того, як працює планувальник, та які процеси відбуваються у процесі планування неможливо приступати до роботи над інтеграцією нового алгоритму. Тому однією з проблем, з якою зіштовхнулися в процесі вивчення роботи планувальника та інтеграції нового алгоритму до складу планувальника Maui, є недостатній опит розроблення програмного забезпечення під управлінням Unix системою. Також, розробники планувальника залишили дуже не велику кількість коментарів до коду, що, в свою чергу, також значно ускладнює вивчення принципів та механізмів роботи планувальника, а також ускладнює його відладку та тестування. Все це потрібно для більш кращого розуміння того, як працює планувальник.

На сьогоднішній час розглянутий алгоритм планування вже реалізований, але у Windows-додатку. Цей Windows-додаток (далі – модель) моделює роботу кластеру та процеси планування у ньому. Він дозволяє провести аналіз ефективності роботи алгоритму при різних умовах, а саме при різних показниках продуктивності ресурсів, різних по складності викання завдань та різній інтенсивності завдань.

Так як алгоритм вже реалізований, то було прийняте рішення піти шляхом портування програмного забезпечення алгоритму до платформи Unix.

Модель роботи кластеру має графічний інтерфейс для взаємодії з користувачем. Отже першою проблемою, з якою довелося зіштовхнутися при інтеграції, є виділення вихідного коду алгоритму з Windows-додатку. Складність вирішення цієї проблеми полягає в тому, що елементи графічного інтерфейсу додатку тісно пов'язані з кодом алгоритму, тому при компіляції вихідного коду алгоритму під платформою Unix виникала велика кількість помилок. За допомогою цих елементів користувачеві виводилася різноманітна інформація щодо процесу планування. У процесі портування ця проблема була вирішена шляхом модифікації коду алгоритму та видалення з нього елементів графічного інтерфейсу Windows-додатку.

При реалізації цього алгоритму у Windows-додатку була використана стандартна бібліотека шаблонів (Standard Template Library; STL) – бібліотека для мови програмування C++, що містить набір узгоджених узагальнених алгоритмів, контейнерів, засобів доступу до їхнього вмісту і різних допоміжних функцій. Процес планування у моделі був реалізований саме за допомогою динамічних контейнерів STL, а саме таких, як vector, set, тощо. В них зберігаються всі елементи у процесі плануванні. Ці контейнери мають власні функції для додавання, видалення, та доступу до елементів контейнеру. Проаналізувавши вихідний код планувальника Maui, стало відомо, що у ньому всі елементи, якими оперує планувальник у процесі планування, зберігаються у звичайних статичних масивах. Другою проблемою є зведення масивів та контейнерів, яка була вирішена шляхом реалізації функцій додавання та видалення елементів зі статичних масивів до динамічних контейнерів і навпаки.

Після аналізу вихідного коду планувальника стало, також, відомо, що завдання та вузли кластеру завдаються у ньому так званими користувальницькими типами даних (структурами). Враховуючи те, що Windows-додаток, який моделює роботу кластеру, являє собою лише модель, то деякі параметри роботи кластеру, завдань та процесу планування задавалися простими типами даних. Отже потрібно звести ці різні типи даних. Це стало третьою проблемою при інтеграції нового алгоритму до складу планувальника. Всі користувальницькі типи даних (структури) будуються на простих типах (рис. 2).

```
typedef struct mjob_t {
  char   Name[MAX_MNAME + 1]; /* job ID */
  char  *AName;   /* alternate name (user specified) */
  char  *RMJID;   /* resource manager job ID */

  int    Index;   /* job table index */

  long   CTime;   /* creation time (first RM report) */
  long   MTime;   /* modification time (any source) */
  long   ATime;   /* access time (most recent RM report) */

  long   StateMTime;

  long   SWallTime;/* duration job was suspended   */
  long   AWallTime;/* duration job was executing   */

  mjckpt_t *Ckpt;  /* checkpoint structure      */

  struct mjob_t *Next;
  struct mjob_t *Prev;

  mjobcache_t C;
  macl_t      RCL[MAX_MACL]; /* required CL   */
  mcred_t     Cred;

  mqos_t     *QReq;

  mres_t     *R;             /* reservation */
  char        ResName[MAX_MNAME];
  /* reservation access list */
  char        RAList[MMAX_JOBRA][MAX_MNAME];
```

Рис. 2. Частина структури даних mjob_t, яка описує завдання

Виходячи з цього, було прийнято рішення, що не потрібно зводити ці типи, а потрібно отримувати доступ до потрібних полів структури, які описуються простими типами даних, через операцію розіменування.

За політику виділення ресурсів у конфігураційному файлі планувальника Maui відповідає параметр NODEALLOCATIONPOLICY (рис. 3).

```
SERVERHOST            localhost
ADMIN1                root
RMCFG[S200]           TYPE=PBS
AMCFG[bank]           TYPE=NONE
RMPOLLINTERVAL        00:00:30
SERVERPORT            42559
SERVERMODE            NORMAL
LOGFILE               maui.log
LOGFILEMAXSIZE        10000000
LOGLEVEL              3
QUEUETIMEWEIGHT       1
BACKFILLPOLICY        FIRSTFIT
RESERVATIONPOLICY     CURRENTHIGHEST
NODEALLOCATIONPOLICY  MINRESOURCE
```

Рис. 3. Приклад можливого лістингу файлу конфігурації «Maui»

Одним зі значень цього параметру може бути значення LOCAL. Це значення надає можливість визвати локально створений алгоритм виділення ресурсів на завдання у каталозі дистрибутиву maui-3.*.*\contrib\nodeallocation\ (рис. 4). Саме у цьому каталозі і буде знаходитись файл вихідного коду з новим алгоритмом планування та виділення ресурсів. Після успішного завершення написання коду алгоритму файл з вхідним кодом буде підключений до файлу MLocal.c у каталозі maui-3.*.*\src\moab\, у цьому ж файлі буде вставлений код відклику алгоритму, а саме у функції MLocalQueueScheduleIJobs.

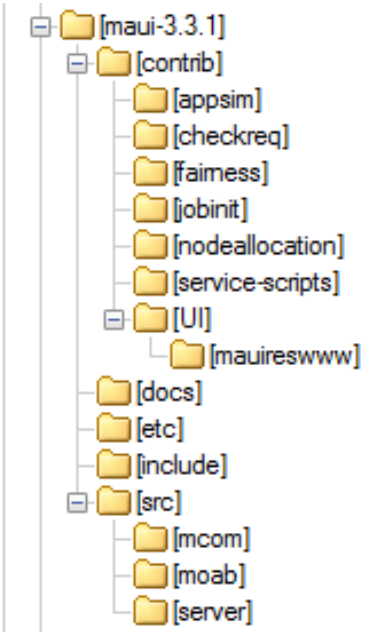

Рис. 4. Структура дистрибутиву «Maui»

Виклик нового алгоритму треба робити в місці, де розробники залиши коментар: «NOTE: insert call to scheduling algorithm here» (рис. 5) [9].

Файл вихідного коду планувальника Maui MLocal.c відповідає за роботу локально створених алгоритмів планування та політик виділення ресурсів під завдання.

```
int MLocalQueueScheduleIJobs(

  int    *Q,
  mpar_t *P)

  {
  mjob_t *J;

  int     jindex;

  if ((Q == NULL) || (P == NULL))
    {
    return(FAILURE);
    }

  /* NOTE:  insert call to scheduling algorithm here */

  for (jindex = 0;Q[jindex] != -1;jindex++)
    {
    J = MJob[Q[jindex]];

    /* NYI */

    DBG(7,fSCHED) DPrint("INFO:     checking job '%s'\n",
      J->Name);
    }  /* END for (jindex) */

  return(FAILURE);
  }  /* END MLocalQueueScheduleIJobs() */
```

Рис. 5. Функції MLocalQueueScheduleIJobs

Компіляція вихідного коду планувальника з інтегрованим алгоритмом під платформу Unix виконується компілятором g++, який призначений для мови програмування C++ так, як вихідний код алгоритму має елементи STL, які компілятор gcc для мови програмування C не підтримує.

Висновки

Таким чином, була розглянута програмна база побудови кластеру, а саме такі програмні продукти, як менеджер ресурсів Torque та планувальник Maui, їх структура та загальні положення щодо механізмів їх роботи. Також був розглянутий алгоритм резервування та планування Maui.

Був проаналізований вихідний код планувальника Maui, його особливості та функціонал.

Були висвітлені проблеми, пов'язані зі специфікою інтеграції нового алгоритму планування до складу Maui, а саме проблеми пов'язані з програмною реалізацією алгоритму під платформою Unix, проблеми відладки та тестування планувальника, а також побудови вхідних даних для вирішення задачі про найменше покриття, яка покладена в основу нового алгоритму, який інтегрується до ПЗ планувальника Maui.

Доведена можливість включення нових алгоритмів планування у планувальник завдань Maui, що дає сприятимемо підвищенню продуктивності роботи кластерних систем.

Список літератури

1. *Мінухін С.В. Дослідження методів локальних планувальників ресурсів та їх модифікації в ГРІД-системах./С.В. Мінухін, О.В. Мєзєнцев. // Системи обробки інформації. – 2012. – Вип. 4 (102). Т.1. – С. 42–48.*
2. *Система пакетной обработки заданий Torque: Руководство пользователя. Т – Платформы, 2008. [Електронний ресурс]. – Режим доступу до ресурсу: hpc.ssau.ru/files/doc/torque_manual.pdf.*
3. *Adaptive Computing – Documentation / Maui Administrator's Guide. Version 3.2 [Електронний ресурс]. – Режим доступу до ресурсу: http://www.adaptivecomputing.com/resources/docs/maui/index.php.*
4. *Коваленко В.Н. Полигон Грид в ИПМ РАН и разработка методов управления ресурсами в глобальной среде / В.Н. Коваленко, Д.А. Корягин. // X конференция представителей региональных научно-образовательных сетей, RELARN-2003, Санкт-Петербург, 16-20 июня 2003.// Сборник тезисов докладов, Тровант, 2003. – С. 214–216.*
5. *Adaptive Computing – Documentation / Maui Administrator's Guide. Version 3.2 [Електронний ресурс]. – Режим доступу до ресурсу: http://www.adaptivecomputing.com/resources/docs/maui/index.php*
6. *Коваленко В.Н. Использование алгоритма BACKFILL в ГРИД / В.Н. Коваленко., Д.А. Семячкин. [Електронний ресурс]. – Режим доступу до ресурсу: http://gridclub.ru/library/publication.2004-12-27.6965428621/publ_file/.*
7. *Листровой С.В. Модель и подход к планированию распределения ресурсов в гетерогенных Грид-системах /С.В. Листровой, С.В. Минухин. / Международный научно-технический журнал «Проблемы управления и информатики». – 2012. – № 5. – С. 120–133./*
8. *The DataGrid Projec. [Електронний ресурс]. – Режим доступу до ресурсу: http://eu-datagrid.web.cern.ch/.*
9. *Job scheduling strategies for parallel processing: IPDPS 2000 workshop, JSSPP 2000, Cancun, Mexico, May 2000, Proceedings[Електронний ресурс]. – Режим доступу до ресурсу: http://www.informatik.uni-trier.de/~ley/db/conf/jsspp/jsspp2000.html.*
10. *Adaptive Computing – Documentation / TORQUE Administrator's Guide. Version 3.0.3 [Електронний ресурс]. – Режим доступу до ресурсу: http://www.adaptivecomputing.com/resources/docs/torque/3-0-3/index.php.*

Надійшла до редколегії 9.03.2013

Рецензент: д-р техн. наук, проф. В.С. Харченко, зав. кафедри комп'ютерних систем і мереж Національного аерокосмічного університету ім. М.Є. Жуковського «ХАІ», Харків, Україна.

ИССЛЕДОВАНИЕ ПЛАНИРОВЩИКА РЕСУРСОВ MAUI И ВОЗМОЖНОСТЕЙ ИНТЕГРАЦИИ В ЕГО СОСТАВ НОВЫХ АЛГОРИТМОВ

С.В. Минухин, А.В. Мезенцев

Проанализированы системы управления распределенными вычислениями и методы планирования ресурсов в кластерных системах. Проанализированы алгоритмы планирования ресурсов на примере планировщика Maui. Исследованы исходный код планировщика ресурсов. Приведены и проанализированы особенности программной реализации нового алгоритма на основе метода покрытия в составе планировщика Maui. Исследовано расширение возможностей планировщика Maui путем интеграции в программное обеспечение новых планировщиков для повышения эффективности его работы.

***Ключевые слова**: ресурсы, кластер, система управления пакетной обработкой заданий, PBS, Torque, Maui, планировщик, наименьшее покрытие.*

INVESTIGATION OF RESOURCE SCHEDULER MAUI AND POSSIBILITIES OF INTEGRATION INTO ITS STRUCTURE NEW ALGORITHMS

S.V. Minukhin, O.V. Miezientsev

The system of management of distributed computing and methods of scheduling resources in the cluster systems. Analyzed algorithms resource planning example scheduler Maui. Investigated the source resource scheduler. Presented and analyzed features a software implementation of a new algorithm based on the minimal cover method in the scheduler Maui. Investigated empoverment Maui through the integration of new software schedulers to improve its efficiency.

***Keywords:** resources, cluster, the control batch system, PBS, Torque, Maui, scheduler, minimal cover.*