УДК 004.382.2, 004.457
DOI: 10.15827/0236-235X.122.295-302
Дата подачи статьи: 30.10.17
2018. Т. 31. № 2. С. 295–302

СИСТЕМА РАСПАРАЛЛЕЛИВАНИЯ НАГРУЗКИ НА РЕСУРСЫ ЭВМ

Е.В. Пальчевский [1], *аспирант, teelxp@inbox.ru*
А.Р. Халиков [1], *к.ф.-м.н., доцент, khalikov.albert.r@gmail.com*

[1] *Уфимский государственный авиационный технический университет, ул. Карла Маркса, 12, г. Уфа, 450008, Россия*

Данная статья посвящена распараллеливанию нагрузки на физические ресурсы ЭВМ за счет алгоритма, основанного на цепях Маркова. Разработан аппаратно-программный модуль IDLP, предназначенный для распараллеливания вычислительных процессов на физические ресурсы вычислительного кластера. Показана нагрузочная зависимость физических ресурсов от запускаемых на вычислительном кластере задач. Обоснована целесообразность применения аппаратно-программного модуля IDLP. Созданный аппаратно-программный модуль позволяет эффективно справляться с переносами вычислительных процессов, а также предотвращает перегруженность системы. Одним из элементов модуля является интуитивно понятный веб-интерфейс, позволяющий удаленно управлять IDLP как со смартфона, так и с персонального компьютера.

В статье описан функционал разработанного аппаратно-программного модуля: старт, стоп, перезапуск, установка значения нагрузки ядра. Показана схема данного модуля, на которой представлено также равномерное распределение (распараллеливание) вычислительных процессов по физическим и логическим ядрам ЭВМ с последующим снижением нагрузки на центральный процессор.

В работе приведены результаты тестирования аппаратно-программного модуля, в ходе которого подтверждено снижение загруженности вычислительных ресурсов физического сервера. Как следствие – повышение производительности. Разработанное решение позволяет не только повысить производительность в 1,72 раза (на конкретном используемом оборудовании, в остальных случаях нагрузка может отличаться), но и параллельно запускать сложные и ресурсоемкие вычислительные процессы без нарушения работоспособности ЭВМ.

Ключевые слова: *распараллеливание, процесс, обработка данных, вычислительные кластеры, вычислительные процессы.*

Высокопроизводительные технологии находят все более широкое применение в IT-сфере [1]. Это обусловлено технологическим прорывом в области информационных технологий, позволяющим ускорить решение вычислительных задач за счет параллельных вычислений [2]. В связи с увеличением сложности и объемов поставленных задач возникает проблема параллельных и сбалансированных вычислений на кластерных технологиях. Зачастую вычислительные кластеры размещаются в *центрах обработки данных* (ЦОД). Как правило, там устанавливаются операционные системы семейств UNIX, Linux, Windows, Solaris, Amiga и других. Однако практически для каждого вычислительного кластера существует проблема повышенной загруженности из-за неоптимального распределения процессов по физическим и логическим ядрам.

Одним из главных требований современных информационных технологий является высокая производительность, и распределенная нагрузка способствует исключению перегруженности кластера при параллельных вычислениях, а также при других ресурсоемких процессах. Вопросами распределенных вычислений занимается широкий круг исследователей, о чем свидетельствуют многочисленные публикации. Например, автор [3] рассматривал преобразование программ, выполняющее круговой сдвиг операторов тела цикла, с последующим использованием для распараллеливания. Авторы [4] исследовали параллельные варианты метода динамического программирования для задачи о сумме подмножеств. В [5] получены оценки эффективности и ускорения на примере решения систем алгебраических уравнений методом циклической редукции на HPC-кластере. Автор [6] распараллеливал решения задач линейной алгебры с последующим достижением минимизации числа используемых внешних каналов. В [7] предложен алгоритм выделения контуров на изображениях на основе кластеризации с возможностью распараллеливания. Авторы [8] разработали подход, позволяющий ускорять выполнение запросов класса OLAP в сотни раз, а в [9] реализован параллельный алгоритм поиска локально похожих последовательностей временного ряда для ускорителей на базе архитектуры Intel MIC. В [10] предложено условие, с помощью которого произведено доказательство теоремы о сходимости конечнообразных аппроксимаций метода регуляризации Тихонова к точному решению регуляризованной задачи.

В связи с тем, что вычисления становятся все более ресурсоемкими, растет нагрузка на центральные процессоры. Чрезмерная загруженность приводит к снижению производительности вычислительного кластера и повышению энергопотребления. Поэтому актуальной задачей является повышение производительности и равномерное распределение вычислительных процессов по физическим и логическим ядрам в автоматическом режиме.

В статье описана разработка аппаратно-программного модуля IDLP (англ. I distribute the load

processes) для равномерного распределения нагрузки вычислительного кластера с последующим выводом данных в веб-интерфейс. Данный модуль позволяет повышать производительность вычислительного кластера путем снижения его загруженности, а также уменьшает энергопотребление системы в целом. Представлен обзор существующих решений (кластеров) для выполнения каких-либо ресурсоемких задач. Показаны преимущества и научная новизна предлагаемого решения, а также архитектура (структура) и разработка системы. Представлены результаты апробации аппаратно-программного модуля IDLP, в результате которой была выявлена его эффективность и, как следствие, повышенная производительность вычислительного кластера.

Обзор существующих решений

В данном разделе представлен обзор существующих решений – кластеров. Кластер представляет собой группу физических серверов, объединенных при помощи высокоскоростных каналов связи в единый вычислительный ресурс. Существуют следующие виды кластеров [11]:

- программные;
- вычислительные;
- распределения нагрузки;
- отказоустойчивые (высокой доступности);
- распределенные.

Программные кластеры представляют собой логическое объединение физических серверов в единый вычислительный ресурс. Схема работы программного кластера представлена на рисунке 1.

Ненаправленные дуги означают объединение компонентов (сервер бекапов, головной узел и т.д.). Направления от локальной сети означают взаимосвязь вычислительного кластера по локальной сети. Головной сервер отвечает за управление программным кластером и является точкой входа.

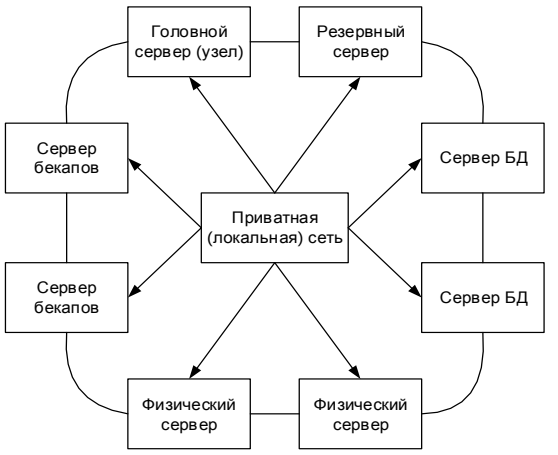

Рис. 1. Схема работы программного кластера

Fig. 1. The software cluster scheme

Резервный сервер представляет собой вычислительный запас, предназначенный для повышения отказоустойчивости: если один из серверов выходит из строя, вместо него запускается резервная ЭВМ. Основными недостатками работы программного кластера являются низкая надежность и невозможность переноса ресурсоемких вычислительных процессов по физическим и логическим ядрам каждого сервера стандартными методами [12].

Вычислительные кластеры представляют собой единый ресурс, предназначенный для ресурсоемких операций по каким-либо вычислениям. Схема работы вычислительного кластера дана на рисунке 2.

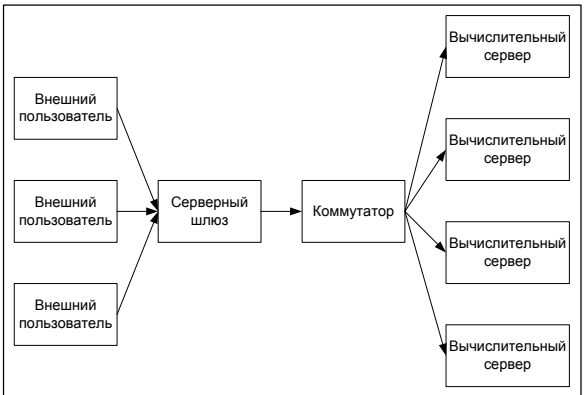

Рис. 2. Схема работы вычислительного кластера

Fig. 2. The computing cluster operation scheme

Направленные дуги означают последовательную связь между выполнениями действий: внешний пользователь подключается к вычислительному кластеру, предварительно соединившись с серверным шлюзом и коммутатором. Основными недостатками вычислительного кластера являются большой промежуток времени взаимодействия между узлами, что (зачастую) приводит к высокой нагрузке на центральные процессоры, и, как следствие, невозможность распараллеливания и переноса вычислительных процессов (ресурсоемких, стандартными методами) по физическим и логическим ядрам каждого сервера.

Кластеры распределения нагрузки (Network Load Balancing, NLB) представляют собой единый вычислительный ресурс, предназначенный для повышения производительности [13]. Производительность достигается при равномерном распределении сетевой нагрузки на все физические серверы. Схема работы NLB показана на рисунке 3.

В данном случае направленные дуги означают соединение балансирующих серверов при помощи локальной вычислительной сети и их объединение в единый ресурс. Недостатком такого кластера является высокая вероятность очереди между взаимодействующими узлами, что приводит к увеличению нагрузки на центральный процессор каждого физического сервера. Распределение нагрузки

Рис. 3. Схема работы NLB

Fig. 3. NLB operation scheme

стандартными методами не дает необходимого эффекта, так как процессы являются ресурсоемкими и затрудняют взаимодействие между собой.

Отказоустойчивые кластеры (высокой доступности) представляют собой единый комплексный ресурс с избыточным числом сетевых узлов [14]. В случае отказа какого-либо кластерного (сетевого) узла избыточные (резервные) физические серверы гарантируют надежность доступности ресурса. Схема работы отказоустойчивых кластеров показана на рисунке 4.

В схеме работы отказоустойчивого кластера направленные дуги означают соединение между различными элементами: пользователь подключается к коммутатору ЛВС, далее происходит запрос на авторизацию к коммутаторам VLAN1 и VLAN2. После успешной авторизации через VLAN1 и VLAN2 происходит подключение к кластерному узлу, а далее к коммутатору сети хранения данных. Между SAN и резервными серверами происходит взаимодействие. Недостатком отказоустойчивого кластера является низкая производительность: невозможно одновременно перемещать процессы и задачи между физическими серверами, тем самым нет возможности распределения (стандартными средствами) процессов между физическими и логическими ядрами на разных серверах.

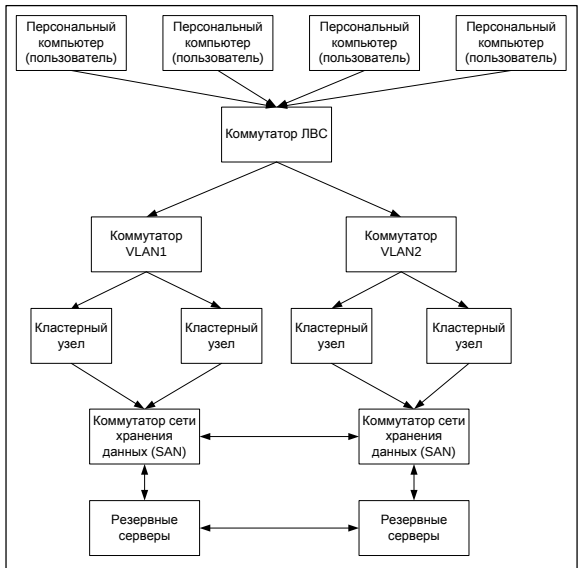

Рис. 4. Схема работы отказоустойчивого кластера

Fig. 4. The failover cluster operation scheme

Распределенные кластеры (GRID-системы) представляют собой единый вычислительный ресурс, реализованный на основе виртуального суперкомпьютера, который представлен в виде кластеров. Схема работы распределенного кластера показана на рисунке 5.

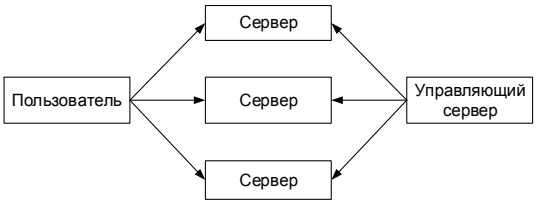

Рис. 5. Схема работы распределенного кластера

Fig. 5. The distributed cluster operation scheme

Направленные дуги означают последовательную направленность между элементами: работа пользователя с распределенным кластером как напрямую, так и через управляющий сервер. Недостатками данной системы являются низкая доступность каждого узла, что не дает гарантии работоспособности в настоящий момент времени, а также невозможность распараллеливания процессов между кластерами, физическими и логическими ядрами. Подобный эффект объясняется низкой производительностью при высоких нагрузках.

Общим недостатком перечисленных решений является невозможность корректного распределения нагрузки по физическим и логическим ядрам стандартными методами. Предлагаемое решение IDLP позволяет равномерно распределять нагрузку по физическим и логическим ядрам процессора, тем самым снижая риск перегрузки всей системы.

Преимущества и научная новизна

Научная новизна аппаратно-программного модуля IDLP заключается в переносе вычислительных процессов алгоритмом, основанным на цепях Маркова, между ядрами всех физических серверов кластера. Схема работы данного алгоритма представлена на рисунке 6. Его решение отличается от вышеприведенных корректным и равномерным распределением нагрузки на физические ресурсы ЭВМ, а также оснащением необходимым функционалом по удаленному управлению. Основные преимущества предлагаемого решения:

– возможность работы в автоматизированном режиме без участия человека: можно запустить алгоритм один раз, а далее старт будет происходить автоматически;

– проверка загруженности как всего вычислительного кластера, так и каждого физического сервера по отдельности;

– равномерное распределение нагрузки на вычислительном кластере с использованием цепей Маркова;

Рис. 6. Схема работы модуля на основе цепей Маркова

Fig. 6. The operation scheme of the Markov chain module

– удобное управление через веб-интерфейс со смартфона и персонального компьютера.

При создании нового процесса происходят проверка загруженности вычислительного кластера, а также сравнение порогового числа (лимита одновременно запущенных задач) с количеством запущенных заданий. После проверки запускается алгоритм цепи Маркова, который подключается к кластеру, а затем и к каждому физическому серверу. После подключения начинается выполнение алгоритма цепи Маркова. В данном случае цепи Маркова доработаны из периодического состояния цепи и представляют собой перенос случайных загружаемых ресурсы ЭВМ процессов по ядрам каждого физического сервера кластера. Соответственно, если $d(j) > 1$, то состояние является периодическим. Если $d(j) = 1$, то состояние является апериодическим. Переменная $d(j)$ представляет собой состояние периодическое либо апериодическое. Таким образом, если периоды состояний совпадают, то $(i \leftrightarrow j) \Rightarrow (d(i) = d(j))$.

Количество физических и логических ядер в кластере необходимо для выборки диапазона серверов с последующим распределением нагрузки на ресурсы ЭВМ и определяется следующим образом:

$$Y = S \cdot (P \cdot FL), \quad (1)$$

где S – количество физических серверов; P – количество процессоров в одной ЭВМ; FL – количество физических и логических ядер на одной выделенной машине. Начальное распределение загруженности необходимо для первостепенного уменьшения нагрузки на ресурсы ЭВМ и представлено в виде

$$r = \{Y, Z\} = (P_k \cdot Y) \cdot Z, \quad (2)$$

где Y – общее количество ядер в кластере; Z – загруженность кластера; P_k – количество входящих заявок, представляемых в виде вычислительных процессов.

Также разработан механизм учета и исправления возможных ошибок при одновременном переносе нескольких вычислительных процессов:

$$E = \frac{n}{Y} = \frac{\mu}{\sum_{i=0}^{n} \theta}, \quad (3)$$

где E – вероятность ошибок; n – количество переносимых процессов с возможной ошибкой; Y – сумма ядер в кластере; μ – время обработки каждого вычислительного процесса; $\sum_{i=0}^{n} \theta$ – стремящийся к разгрузке вычислительный поток. В случае некорректного переноса вычислительного процесса на другое ядро может возникнуть его перегрузка. Соответственно, снизится производительность физического сервера. Таким образом, данный механизм учета (3) снижает риски некорректных переносов вычислительных процессов в режиме реального времени.

После выполнения алгоритма повторная проверка загруженности процессоров кластера запускается 1 раз в 10 минут. Данное значение актуально для кластера, на который была произведена инсталляция алгоритма. В остальных случаях значение может отличаться. Таким образом, каждые 10 минут алгоритм срабатывает по заданию из Cron, предназначенного для выполнения задаваемых команд (например, запуск или перезагрузка) в определенное время. Если поставить повторную проверку алгоритма более 10 минут, то увеличивается риск перегрузки системы из-за возрастания потребления вычислительных ресурсов и может произойти их утечка. В случае, если установить алгоритм на проверку менее 10 минут, система не будет успевать проверять каждый вычислительный процесс. Если нагрузка на одно из ядер превышает 50 %, ресурсоемкий процесс переносится на другие ядра данного физического сервера (3). Если все остальные ядра физического сервера загружены более чем на 50 %, перенос происходит на другие физические серверы кластера. При загрузке всех физических и логических ядер кластера на 50 % и более отправляется уведомление в веб-интерфейс о том, что необходимо изменить текущий лимит нагрузки. Представленный алгоритм позволяет снижать загруженность вычислительного кластера и повышать его производительность.

Архитектура и разработка аппаратно-программного модуля IDLP

IDLP – модуль, предоставляющий возможность параллельного переноса процесса с одного ядра на другое за счет применения сегментативной оптимизации, которая заключается в блоковом объединении и переносе вычислительных процессов как массово (блоком), так и по одному. Данные блоки формируются по убыванию загруженности ресур-

Таблица 1

Описание выполняемых команд IDLP

Table 1

The description of running IDLP commands

Команда	Цель выполнения	Условие работы	Полезность при выполнении
Старт	Запуск модуля: автоматический и ручной (из web-части, в консольном режиме)	Автоматический запуск	Распределение нагрузки по ядрам
Стоп	Остановка модуля: автоматическая и ручная (из web-части, в консольном режиме)	Ручное выполнение команды	Уведомление о загруженности ядер в web-интерфейс с последующими рекомендациями
Перезапуск	Перезагрузка модуля из web-части и консоли	Ручное выполнение команды	Меньшее потребление ресурсов
Установка значения нагрузки ядра	Задание собственного значения лимита загрузки каждого ядра	Достижение лимитированного значения, заданного из web-части. Значение по умолчанию: 50 %	Распределение процессов при достижении заданных лимитов

сов каждым процессом. Например, если на физическом сервере вычислительные процессы нагружают центральный процессор на 30–50 %, то модуль объединяет их в один блок. Модуль также предназначен для снижения загруженности вычислительного кластера, что способствует повышению его производительности.

Модуль предоставляет веб-интерфейс для просмотра результатов работы (рис. 7). Вывод пинга от Уфы до кластера в Москве предназначен для проверки сетевых маршрутов. Данные города взяты из-за расположения вычислительного кластера в одном из ЦОД г. Москвы, а тестирование удаленного веб-интерфейса проводилось из Уфы. Соответственно, значение пинга в норме варьируется в диапазоне от 20 до 28. Повышение пинга, например, от 40 и более говорит о проблеме на одном из промежуточных узлов сетевого канала.

Основной функционал:

– проверка каждого ядра на процент загруженности при создании процесса;

– перенос процессов между физическими и логическими ядрами по всему кластеру: цепи Маркова выявляют менее загруженные ядра с последующей миграцией вычислительного процесса с более загруженного ядра;

– ручной ввод условий проверки (через веб-интерфейс): периодичность, лимит нагрузки на физические и логические ядра; по умолчанию лимит нагрузки – 50 %, период проверки – 1 секунда;

– вывод средней нагрузки всей ЭВМ в веб-интерфейс и сохранение результатов в БД MySQL.

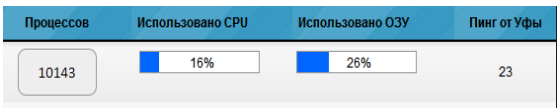

Рис. 7. Вывод количества процессов, загруженности физического сервера и замеряемого ping-модулем IDLP

Fig. 7. Output of the number of processes, load of a physical server and ping-measured IDLP module

Основной принцип работы программного модуля IDLP продемонстрирован на рисунке 8.

Рис. 8. Схема работы IDLP

Fig. 8. IDLP operation scheme

Разработанные функции модуля IDLP, выполняемые с помощью удаленного клиента и консольной оболочки, показаны в таблице 1.

Потребление ОЗУ при деактивированном состоянии модуля IDLP не происходит. Представленные команды позволяют управлять вычислительными процессами с последующим распределением между физическими и логическими ядрами для распараллеливания нагрузки.

Результаты потребляемых ресурсов IDLP в суточном эквиваленте представлены в таблице 2.

Суммарное потребление вычислительных ресурсов аппаратно-программным модулем IDLP: центральный процессор – 0,054 %, оперативная память – 0,00035 %, твердотельный накопитель – 0,002 %.

Таблица 2

Потребление ресурсов при запуске разработанных команд в модуле IDLP

Table 2

Resource consumption when running developed commands in the IDLP module

Команда	Активное состояние	Потребление CPU, %	Потребление ОЗУ, %	Нагрузка на SSD, %
Старт	Да	0,025	0,0001	0,001
Стоп	Да	0	0	0
Перезапуск	Да	0,029	0,00025	0,001
Установка значения нагрузки ядра	Да	0	0	0

Суммарное потребление вычислительных ресурсов разработанным аппаратно-программным модулем IDLP низкое. Таким образом, представленный функционал модуля IDLP позволяет достаточно эффективно бороться с перегруженностью вычислительного кластера.

Тестирование IDLP

После подготовки алгоритма и написания исходного кода аппаратно-программного модуля IDLP необходимо произвести его тестирование для получения значений загруженности системы до и после активации модуля. Результаты представлены в таблицах 3 и 4. С каждым днем автоматически запускались вычислительные процессы на каждом физическом сервере кластера. Данные процессы представляют собой запущенные виртуальные машины (VDS), программные продукты и другие. Запуск данных процессов необходим для более тщательного тестирования разработанного модуля. Тестирование в течение десяти дней объясняется тем, что многие процессы на VDS дают весомую нагрузку только после некоторого промежутка времени. Таким образом, нагрузка возрастала и количество перебросов вычислительных процессов увеличивалось. Потребление ресурсов измерялось на каждом физическом сервере кластера (команда htop, обеспечивающая вывод текущей нагрузки) и суммировалось в самом модуле. Далее нагрузка выводилась в веб-интерфейс на отдельную страницу в виде графика со значениями.

Средняя нагрузка на физические ресурсы: CPU – 9,5 %, ОЗУ – 0,05 %, SSD – 0,01 %.

Нагрузка на вычислительные ресурсы: центральный процессор – 5,50 %, потребление оперативной памяти – 0,10 %, твердотельный накопитель – 0,15 %.

Сравнение среднемесячных результатов до и после обработки модулем IDLP представлено в таблице 5.

Таблица 3

Проверка нагрузки на ресурсы без активного модуля IDLP

Table 3

Checking resource load without an active IDLP module

День	Произведено перебросов вычислительных процессов, шт.	Лимит ядер, %	Средняя нагрузка на CPU, %	Среднее потребление ОЗУ, %	Средняя нагрузка на SSD, %
1	50	100	5,00	0,01	0,01
2	64		6,00	0,02	
3	78		7,00	0,03	
4	90		8,00	0,04	
5	110		9,00	0,05	
6	120		10,00	0,06	
7	130		11,00	0,07	
8	140		12,00	0,08	
9	150		13,00	0,09	
10	160		14,00	0,10	

Таблица 4

Тестирование нагрузки физических ресурсов при обработке модулем IDLP

Table 4

Testing resources load when processing by the IDLP module

День	Произведено перебросов вычислительных процессов, шт.	Лимит ядер, %	Средняя нагрузка на CPU, %	Среднее потребление ОЗУ, %	Средняя нагрузка на SSD, %
1	350	50	1	0,06	0,11
2	650		2	0,07	0,12
3	950		3	0,08	0,13
4	1250		4	0,09	0,14
5	1550		5	0,10	0,15
6	1850		6	0,11	0,16
7	2150		7	0,12	0,17
8	2450		8	0,13	0,18
9	2750		9	0,14	0,19
10	3050		10	0,15	0,20

Таблица 5

Анализ средней нагрузки на физические ресурсы в течение 10 дней (%)

Table 5

Analysis of the average load on physical resources for 10 days (%)

Этап	Нагрузка на CPU	Потребление ОЗУ	Нагрузка на SSD
До обработки	9,50	0,05	0,01
После обработки	5,50	0,10	0,15

Разница средней загруженности вычислительных ресурсов:

– центральный процессор: после активизации модуля IDLP повышается в 1,72 раза;

– потребление оперативной памяти: обрабатываемость программным модулем IDLP повышает объем занятости ОЗУ в 2 раза;

– твердотельный накопитель (SSD): нагрузка повышается в 15 раз.

Таким образом, нагрузка на центральный процессор снизилась в 1,72 раза. Подобный рост производительности на процессорах Intel Xeon 5670 позволяет параллельно запускать более ресурсоемкие вычислительные процессы, тем самым ускоряя выполнение поставленной задачи. Потребление ОЗУ, а также нагрузка на твердотельный накопитель SSD повышаются, но незначительно: данное повышение не повлияет на работоспособность вычислительного кластера. В одном из ЦОД Москвы были опробованы аналогичные стандартные методы распределения нагрузки, показавшие недостаточную эффективность: средняя нагрузка на центральный процессор приравнивается к 35 %. Тестирование разработанного аппаратно-программного модуля IDLP в боевом режиме проводилось на следующем оборудовании: количество физических серверов – 30; процессор Intel Xeon 5670 (CPU – 60, физических ядер – 360, количество потоков – 720); оперативная память – 960 Гб; дисковая подсистема – RAID 10 (Intel S3710 SSDSC2BA012T401 800 Гб каждый); внешний сетевой канал – 20 Гбит/с.

После апробации аппаратно-программный модуль был окончательно инсталлирован в один из московских ЦОД на вычислительный кластер и показывает стабильную работу по сей день. При помощи разработанного модуля было сделано 540 тысяч переносов вычислительных процессов между физическими и логическими ядрами кластера. Средняя загруженность центральных процессоров вычислительного кластера не превышала 7 %.

Заключение

Таким образом, проанализирована и исследована рабочая среда физического сервера и кластера с последующим повышением производительности для стабильной работы ЭВМ.

В ходе проведенных исследований и разработок был реализован аппаратно-программный модуль, обеспечивающий распределение нагрузки по физическим и логическим ядрам как одного физического сервера, так и всего кластера. Обоснована целесообразность применения аппаратно-программного модуля IDLP, предложена методика снижения нагрузки на физические ресурсы ЭВМ. Проведена апробация модуля IDLP, позволяющая определить эффективность данной разработки. Средняя нагрузка на центральный процессор варьируется от 5,5 % до 9,5 %. Потребление оперативной памяти колеблется в диапазоне от 0,05 % до 0,1 %. Нагрузка на твердотельный накопитель изменяется от 0,01 % до 0,15 %.

Разработанная система не только повышает производительность всей рабочей системы, но и предоставляет возможность параллельного запуска сложных и ресурсоемких задач без нарушения рабочей среды ЭВМ.

Литература

1. Соколинская И.М., Соколинский Л.Б. Параллельная реализация следящего алгоритма для решения нестационарных задач линейного программирования // Вестн. ЮУрГУ. Сер.: Вычислительная математика и информатика. 2016. Т. 5. № 2. С. 15–29.

2. Иванова Е.В., Соколинский Л.Б. Декомпозиция операций пересечения и соединения на основе доменно-интервальной фрагментации колоночных индексов // Вестн. ЮУрГУ. Сер.: Вычислительная математика и информатика. 2015. Т. 4. № 1. С. 44–56.

3. Shteinberg O.B. Circular shift of loop body – programme transformation, promoting parallelism // Вестн. ЮУрГУ. Сер.: Математическое моделирование и программирование. 2017. Т. 10. № 3. С. 120–132 (англ.).

4. Посыпкин М.А., Тант Син Си Ту О распараллеливании метода динамического программирования для задачи о ранце. Intern. Jour. of Open Information Technologies. 2017, vol. 5, no. 7, pp. 1–5 (рус.).

5. Овчаренко О.И. Об одном подходе к распараллеливанию метода циклической редукции для решения систем уравнений с трехдиагональной матрицей // Современные тенденции развития науки и производства: сб. матер. III Междунар. практич. конф. Кемерово, 2016. С. 152–155.

6. Пелипец А.В. Распараллеливание итерационных методов решения систем линейных алгебраических уравнений на реконфигурируемых вычислительных системах // Суперкомпьютерные технологии (СКТ-2016): матер. 4 Всерос. науч.-технич. конф. 2016. С. 194–198.

7. Белим С.В., Кутлунин П.Е. Выделение контуров на изображениях с помощью алгоритма кластеризации // Компьютерная оптика. 2015. Т. 39. № 1. С. 119–124.

8. Иванова Е.В., Соколинский Л.Б. Методы параллельной обработки сверхбольших баз данных с использованием распределенных колоночных индексов // Программирование. 2017. № 3. С. 3–21.

9. Мовчан А.В., Цымблер М.Л. Параллельный алгоритм поиска локально похожих подпоследовательностей временного ряда для ускорителей на базе архитектуры INTEL MIC // Суперкомпьютерные дни в России: тр. Междунар. конф. М., 2015. С. 332–343.

10. Танана В.П., Бельков С.И. Конечноразностная аппроксимация метода регуляризации А.Н. Тихонова N-го порядка // Вестн. ЮУрГУ. Сер.: Вычислительная математика и информатика. 2015. Т. 4. № 1. С. 86–98.

11. Цымблер М.Л., Мовчан А.В. Обнаружение подпоследовательностей во временных рядах // Открытые системы. СУБД. 2015. № 2. С. 42–43.

12. Пальчевский Е.В., Халиков А.Р. Автоматизированная

система обработки данных в UNIX-подобных системах // Программные продукты и системы. 2017. Т. 30. № 2. С. 227–234. DOI: 10.15827/0236-235X.030.2.227–234.

13. Пальчевский Е.В., Халиков А.Р. Техника инструментирования кода и оптимизация кодовых строк при моделировании фазовых переходов на языке C++ // Тр. ИСА РАН. Т. 27. 2015. № 6. С. 87–96.

14. Пальчевский Е.В., Халиков А.Р. Равномерное распределение нагрузки аппаратно-программного ядра в UNIX-системах // Тр. ИСА РАН. Т. 28. 2016. № 1. С. 93–102.

Software & Systems
DOI: 10.15827/0236-235X.122.295-302

Received 30.10.17
2018, vol. 31, no. 2, pp. 295–302

THE SYSTEM FOR PARALLELIZING THE LOAD ON COMPUTER RESOURCES

E.V. Palchevsky [1], *Postgraduate Student, teelxp@inbox.ru*
A.R. Khalikov [1], *Ph.D. (Physics and Mathematics), Associate Professor, khalikov.albert.r@gmail.com*

[1] *Ufa State Aviation Technical University, K. Marks St. 12, Ufa, 450008, Russian Federation*

Abstract. The article is devoted to parallelization of the load on computer physical resources using an algorithm based on Markov chains. The IDLP hardware module (I distribute the load processes) is designed to parallelize computation processes to physical resources of a computing cluster. The paper shows load dependence of physical resources on the tasks being run on a computing cluster. It also substantiates the expediency of using the IDLP hardware-software module.

The created hardware and software module allows efficiently handling transfers of computing processes and prevents system congestion. One of the module elements is a user-friendly interface that allows remote managing IDLP from both a smartphone and a personal computer. The load parallelization system consists of three stages. The first one is algorithm development, the second one is technical implementation, the third one is hardware-software module testing.

At the first stage, the authors present the functional of the developed hardware and software module, which includes start, stop, restart, set the kernel load value. The paper presents the scheme of the developed hardware and software module.

At the second stage, a source code fragment (in Python) is responsible for uniform distribution (parallelization) of computational processes on physical and logical computer cores, followed by decreasing load on a central processor.

At the third stage, the authors test the hardware and software module. The testing confirmed the decrease in the workload of physical server computing resources, and, as a result, it led to increased productivity.

The developed solution not only raises productivity by factor of 1.72 (on the equipment used; the load may differ in other cases), but also allows parallel running of complex and resource-intensive computing processes without disrupting computer performance.

Keywords: parallelization, process, data processing, computing clusters, computing process.

References

1. Sokolinskaya I.M., Sokolinsky L.B. Implementation of parallel pursuit algorithm for solving unstable linear programming problems. *Vestn. YuUrGU. Ser.: Vychislitelnaya matematika i informatika* [Bulletin of South Ural State Univ. Series: Computational Mathematics and Software Engineering]. 2016, vol. 5, no. 2, pp. 15–29 (in Russ.).

2. Ivanova E.V., Sokolinsky L.B. Decomposition of intersection and join operations based on the domain-interval fragmented column indices. *Vestn. YuUrGU. Ser.: Vychislitelnaya matematika i informatika* [Bulletin of South Ural State Univ. Series: Computational Mathematics and Software Engineering]. 2015, vol. 4, no. 1, pp. 44–56 (in Russ.).

3. Shteinberg O.B. Circular shift of loop body – programme transformation, promoting parallelism. *Vestn. YuUrGU. Ser.: Matematicheskoe modelirovanie i programmirovanie* [Bulletin of South Ural State University. Series: Mathematical Modeling, Programming & Computer Software]. 2017, vol. 10, no. 3, pp. 120–132 (in Russ.).

4. Posypkin M.A., Si Tu Tant Sin. On the parallelization of dynamic programming method for knapsack problem. Intern. Jour. of Open Information Technologies. 2017, vol. 5, no. 7, pp. 1–5 (in Russ.).

5. Ovcharenko O.I. One approach to parallelizing the cyclic reduction method for solving systems of equations with a tridiagonal matrix. *Sovremennye tendentsii razvitiya nauki i proizvodstva: sb. materialov III Mezhdunar.-praktich. konf.* [Proc. 3rd Int. Pract. Conf. on Modern Trends in the Development of Science and Production]. 2016, pp. 152–155 (in Russ.).

6. Pelipets A.V. Parallelization of iterative methods for solving systems of linear algebraic equations on reconfigurable computing systems. *Superkompyuternye tekhnologii (SKT-2016): Materialy 4-y Vseros. nauch.-tekhnich. konf.* [Proc. 4th All-Russ. Sci. and Pract. Conf. Supercomputer Technologies]. Rostov-na-Donu: SFedU Publ., 2016, pp. 194–198 (in Russ.).

7. Belim S.V., Kutlunin P.E. Boundary extraction in images using a clustering algorithm. *Kompyuternaya optika* [Computer Optics]. 2015, vol. 39, no. 1, pp. 119–124 (in Russ.).

8. Ivanova E.V., Sokolinsky L.B. Methods for parallel processing of very large databases using distributed column indexes. *Programmirovanie* [Programming and Computer Software]. 2017, no. 3, pp. 3–21 (in Russ.).

9. Movchan A.V., Tsymbler M.L. A parallel algorithm for searching locally similar subsequences of a time series for accelerators based on the INTEL MIC architecture. *Superkompyuternye dni v Rossii: Tr. mezhdunar. konf.* [Proc. Int. Conf. Supercomputer days in Russia]. Moscow, Lomonosov MSU Publ., 2015, pp. 332–343 (in Russ.).

10. Tanana V.P., Belkov S.I. The finite difference approximation for the Tikhonov regularization method of the n-th order. *Vestn. YuUrGU. Ser.: Vychislitelnaya matematika i informatika* [Bulletin of South Ural State Univ. Series: Computational Mathematics and Software Engineering]. 2015, no. 4, vol. 1, pp. 86–98 (in Russ.).

11. Tsymbler M.L., Movchan A.V. Detection of subsequences in time series. *Otkrytye sistemy. SUBD* [Open Systems J.]. 2015, no. 2, pp. 42–43 (in Russ.).

12. Palchevsky E.V., Khalikov A.R. Automated data processing system in UNIX-like systems. *Programmnye produkty i sistemy* [Software & Systems]. 2017, vol. 30, no. 2, pp. 227–234 (in Russ.).

13. Palchevsky E.V., Khalikov A.R. Technique the instrumentation a code and optimization of code lines in modeling phase transitions on the programming language C++. *Trudy ISP* RAN [Proc. of ISP RAS]. 2015, vol. 27, no. 6, pp. 87–96 (in Russ.).

14. Palchevsky E.V., Khalikov A.R. Uniform distribution of hardware-software kernel loading in UNIX-SYSTEMS. *Trudy ISP RAN* [Proc. of ISP RAS]. 2016, vol. 28, no. 1, pp. 93–102 (in Russ.).