Min_c: стратегия неоднородной концентрации задач для энергосберегающих компьютерных расписаний [1]

[1]*Ф. Армента-Кано <armentac@cicese.edu.mx>*
[1]*А. Черных <chernykh@cicese.mx>*
[1]*Х.М. Кортес-Мендоза <jcortes@cicese.edu.mx>*
[2]*Р. Яхьяпур <ramin.yahyapour@gwdg.de>*
[3]*А.Ю. Дроздов <alexander.y.drozdov@gmail.com>*
[4]*П. Буври <Pascal.Bouvry@uni.lu>*
[4]*Д. Клязович <Dzmitry.Kliazovich@uni.lu>*
[5]*А. Аветисян <arut@ispras.ru>*
[6]*С. Несмачнов <sergion@fing.edu.uy>*
[1]*Исследовательский центр CICESE, Энсенада, Мексика*
[2]*GWDG – Геттенгенский университет, Геттенген, Германия*
[3]*МФТИ, Москва, Россия*
[4]*Люксембургский университет, Люксембург*
[5]*ИСП РАН, Москва, Россия*
[6]*Республиканский университет, Мотневидео, Уругвай*

Аннотация. В этой статье мы описываем энергосберегающие онлайн расписания вычислительных задач и механизмы повышения энергоэффективности, учитывая конфликты использования ресурсов. Мы предлагаем модель оптимизации и новый подход к распределению задач, принимая во внимание типы приложений и их концентрацию. Разнородные задачи, решаемые на процессорах, включают в себя приложения, интенсивно использующие процессоры, диски, устройства ввода-вывода, память, сети и т.д. Когда задачи одного типа назначаются на один и тот же ресурс, они могут создать конфликты при использовании CPU, памяти, диска или сети. Это может привести к деградации общей производительности системы и увеличению потребления энергии. Мы описываем энергетические характеристики приложений, учитывая, что выполнение различных задач по-разному влияет на потребляемую мощность за счет использования разного оборудования. Мы предлагаем нелинейную гибридную модель потребления энергии, которая учитывает потребление энергии отдельных приложений и их комбинации. Мы показываем, что умные стратегии распределения задач могут

[1] Работы выполнены при финансовой поддержке Минобрнауки России (Соглашение № 02.G25.31.0061 12/02/2013).

дополнительно улучшить энергопотребление по сравнению с традиционными подходами. Мы предлагаем алгоритмы консолидации разнородных задач и показываем их эффективность на реальных данных в различных сценариях, используя CloudSim для моделирования облачных вычислений. Мы анализируем несколько алгоритмов планирования в зависимости от типа и объема информации, который они используют. Результаты детального моделирования показывают, что с точки зрения минимизации энергопотребления, стратегия, которая балансирует концентрацию задач различных типов Min_c превосходит другие алгоритмы и стабильна в различных сценариях. Эта стратегия приводит к результатам, которые доминируют почти во всех тестах.

Ключевые слова: энергосберегающие алгоритмы, типы приложений, конфликты использования ресурсов, компьютерные расписания

DOI: 10.15514/ISPRAS-2015-27(6)-23

Для цитирования: Армента-Кано Ф., Черных А., Кортес-Мендоза Х.М., Яхьяпур Р., Дроздов А.Ю., Буври П., Клязович Д., Аветисян А., Несмачнов С. Min_c: стратегия неоднородной концентрации задач для энергосберегающих компьютерных расписаний. Труды ИСП РАН, том 27, вып. 6, 2015 г., стр. 355-380. DOI: 10.15514/ISPRAS-2015-27(6)-23.

1. Введение

Облачные вычисления – это глобально распределенная интернет-среда, предоставляющая разнообразные сервисы для использования вычислительных ресурсов и широко распространенная в государственных и частных организациях.

Поставщики облачных вычислений предлагают свои ресурсы и услуги с гарантией качества обслуживания (QoS). Одной из существенных эксплуатационных затрат провайдеров являются энергетические расходы.

Неэффективное управление ресурсами оказывает прямое негативное воздействие на производительность и стоимость. В распределенных средах часто бывает трудно оптимизировать потребление энергии физическими ресурсами и виртуальными машинами (VM) с различными типами задач (интенсивно использующими процессоры, диски, устройства ввода/вывода, память, сеть и т.д.). Необходимо детальное энергетическое управление ресурсами на всех уровнях выполнения задач для оптимизации их использования и увеличения прибыльности [1].

В этой статье мы предлагаем модель оптимизации, в которой принимаются во внимание различные типы приложений, и алгоритм неоднородной консолидации задач для энергосберегающего планирования. Мы оцениваем эффективность наших алгоритмов на реальных данных в различных сценариях и сравниваем их с известными алгоритмами.

Статья имеет следующую структуру: в разделе 2 приводится обзор литературы и рассматриваются алгоритмы по оптимизации

энергопотребления; в разделе 3 описываются постановка задачи и цели исследования; предлагаемые алгоритмы рассматриваются в разделе 4; в разделе 5 приводятся детали экспериментов; раздел 6 описывает методологию, используемую для анализа результатов; экспериментальные результаты приводятся в разделе 7; и, наконец, в разделе 8 описываются основные выводы и направления будущей работы.

2. Обзор литературы

Энергия, необходимая для работы компьютера, блока питания и систем охлаждения вносит существенный вклад в общий объем операционных затрат. Снижение потребления энергии стало одной из основных целей в промышленности, и задачей в научных исследованиях.

В этом разделе мы кратко обсудим известные энергосберегающие алгоритмы назначения ресурсов для выполнения задач, описанные в литературе.

EMVM - Energy-aware resource allocation heuristics for efficient management [2]. Авторы представили алгоритм назначения ресурсов с использованием динамической консолидации виртуальных машин и описали принципы эффективного использования энергии в среде облачных вычислений. В работе показано, что консолидация ведет к значительному сокращению потребления энергии по сравнению с использованием статического распределения ресурсов. Использована следующая модель энергопотребления:

$$\mathbf{P(u) = k * Pmax + (1 - k) * Pmax * u},$$

где Pmax – это максимальная потребляемая мощность, когда сервер используется полностью; k – доля мощности, потребляемой сервером в режиме ожидания (т.е. 70%); u это загрузка процессора. Общее потребление энергии E определяется следующим образом:

$$\boldsymbol{E} = \int_{\mathbf{t_0}}^{\mathbf{t_1}} \mathbf{P(u(t))dt}.$$

Когда несколько VM используют не все ресурсы, они могут быть объединены на минимальном количестве физических ресурсов. Тогда неработающие узлы могут переключаться в спящий режим, чтобы уменьшить общее потребление энергии.

Рис. 1 описывает энергопотребление в соответствии с описанной моделью, в зависимости от нагрузки процессора.

Рис. 1. Процент энергопотребления в зависимости от загрузки процессора (%).

HSFL – Hybrid Shuffled Frog Leaping алгоритм [3]. Эта схема управления ресурсами не только гарантирует качество обслуживания пользователей (QoS), указанное в соглашении об уровне обслуживания (SLA), но также обеспечивает экономию энергии. Авторы используют технологию миграции VM для консолидации ресурсов. Слабо используемые и неиспользуемые ресурсы переводятся в режим экономии энергии, обеспечивая при этом гарантии качества обслуживания. Авторы считают, что потребление энергии сервером осуществляется почти в линейной зависимости от использования процессора. Кроме того, потребление энергии в ждущем режиме составляет 70% от потребления энергии при полной нагрузке, учитывая энергию, потребляемую при миграции VM. Потребление энергии в заданный момент времени h определяется следующим образом:

$$\mathbf{E(h) = 0.7E_{max}(h) + 0.3Utlz(h)E_{max}(h) + 0.1E_{max}} \sum_{\mathbf{i \in v}} \mathbf{T(i)}.$$

$E_{max}(h)$ – это потребление энергии в режиме полной нагрузки. $Utlz(h)$ —- это средний коэффициент использования процессора за единицу времени, v – это множество мигрирующих виртуальных машин, и $T(i)$ – это время миграции VM_i. Потребление энергии в зависимости от процента использования процессора такое же, как показано на рис. 1.

AETC – Algorithm of Energy-aware Task Consolidation [4]. Авторы предлагают алгоритм консолидации задач для минимизации потребления электроэнергии. Алгоритм работает в центре обработки данных для миграции виртуальных машин, которые назначены на процессоры, находящихся в одной стойке, или на стойках, пропускная способность каналов связи между которыми относительно постоянна. Алгоритм ограничивает использование процессора не выше заданного максимального порога в 70% за счет консолидации задач на виртуальных кластерах. Кроме того, в расходах на электроэнергию учитывается время задержки в сети, необходимое для перемещения задачи в другой виртуальный кластер. Энергопотребление виртуальных машин в состоянии простоя и при передаче их по сети рассматривается как константы. Рис.2 показывает энергопотребление в зависимости от нагрузки процессора,

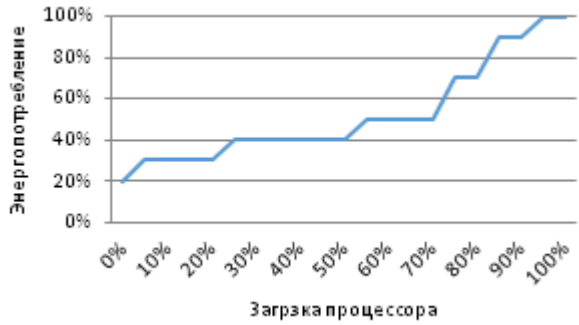

Рис. 2. Энергопотребление в зависимости от загрузки процессора (%).

Эта модель предполагает потребление энергии $E(V_i) = \alpha\ W/s$ в режиме ожидания. Когда загрузка процессора увеличивается, требуется дополнительная энергия β

$$E(Vi) = \begin{cases} \boldsymbol{\alpha\, W/s}, & \boldsymbol{if\ is\ idle} \\ \boldsymbol{\beta + \alpha\, W/s}, & \boldsymbol{if\ 0\% < CPU\ util \leq 20\%} \\ \boldsymbol{3\beta + \alpha\, W/s}, & \boldsymbol{if\ 20\% < CPU\ util \leq 50\%} \\ \boldsymbol{5\beta + \alpha\, W/s}, & \boldsymbol{if\ 50\% < CPU\ util \leq 70\%} \\ \boldsymbol{8\beta + \alpha\, W/s}, & \boldsymbol{if\ 70\% < CPU\ util \leq 80\%} \\ \boldsymbol{11\beta + \alpha\, W/s}, & \boldsymbol{if\ 80\% < CPU\ util \leq 90\%} \\ \boldsymbol{12\beta + \alpha\, W/s}, & \boldsymbol{if\ 90\% < CPU\ util \leq 100\%} \end{cases}$$

Общее потребление энергии виртуальной машиной V_i в течение периода времени $t_0 \sim t_m$ вычисляется по следующей формуле:

$$\mathbf{E_{0,m}(V_i)} = \sum_{\mathbf{t=0}}^{\mathbf{m}} \mathbf{E_t(V_i)}.$$

Для заданного виртуального кластера VC_k, состоящего из n виртуальных машин, потребление энергии в течение периода времени $t_0 \sim t_m$ рассчитывается следующим образом:

$$\mathbf{E_{0,m}(VC_k)} = \sum_{\mathbf{i=0}}^{\mathbf{n}} \mathbf{E_{0,m}(V_i)}.$$

CTES – Cooperative Two-Tier Energy-Aware Scheduling [5]. Авторы рассматривают кооперативный двухуровневый подход к планированию задач с регулированием скорости их выполнения, с целью достижения оптимального использования процессора, вместо миграции задач на другие узлы. Используются несколько стратегий планирования с прогнозом выполнения задач для оптимального назначения их на доступные виртуальные машины. Результаты моделирования показывают, что этот подход уменьшает

общее потребление энергии в облаке. Загрузка процессора определена как: $u_i(t) = \frac{ah_i(t)}{mh_i}$, где $ah_i(t)$ выделенная скорость (MIPS) процессора $host_i$ в момент времени t; mh_i - максимальная вычислительная мощность $host_i$. Авторы полагают, что неиспользуемая машина будет выключена немедленно. Таким образом, суммарная мощность процессора определяется как:

$$\mathbf{P} = \begin{cases} \mathbf{P^{static} + P^{dynamic}} & \mathbf{u > 0} \\ \mathbf{0} & \mathbf{o.w} \end{cases}$$

P^{static} – это мощность, потребляемая в течение времени простоя вычислительного узла. Она определяется как $P^{static} = \alpha P^{max}$, где P^{max} – потребляемая мощность при работе с максимальной загрузкой. Загрузка α – это постоянное соотношение между статической мощностью и максимальной мощностью ($0 < \alpha \leq 1$), которое зависит от физических свойств процессора.

Динамическое энергопотребление определяется как:

$$\mathbf{P_i^{dynamic} = \left(P_i^{max} - P_i^{static}\right)\ u_i^{\gamma}(t).}$$

Если система использует мощность P(u), энергопотребление будет $E = \int_0^{t_{min/u}} P(u)dt$, где t_{min} – это время, в течение которого процессор работает на максимальной вычислительной мощности. Доля потребления энергии в результате использования процессора похожа на показаную на рис. 1. Таким образом выполнение инструкций достигается за счет энергопотребления:

$$\mathbf{E = [\alpha + (1 - \alpha)u^{\gamma}]\frac{P^{max}t_{min}}{u}}$$

DVMA – A Decentralized Virtual Machine Migration Approach [6]. Авторы предлагают децентрализованную миграцию виртуальных машин. Они описывают модель системы и ее энергопотребления, включающие вектора загрузки, сбор информации о загрузке, выбор VM для миграции и пункт назначения миграции. Результаты оценки производительности показывают, что такой подход может обеспечить улучшение балансировки нагрузки и меньшее энергопотребление в сравнении с другими стратегиями.

Пусть α будет доля мощности, потребляемая в момент простоя по сравнению с полной загрузкой; θ – доля мощности, потребляемая в момент текущего использования процессора. Энергопотребление вычисляется следующим образом:

$$\mathbf{P_i = \alpha * P_i^{max} + (1 - \alpha) * P_i^{max} * \theta,}$$

где P_i^{max} – потребляемая мощность, когда процессор используется полностью (то есть, достигает 100% загрузки). Энергопотребление, в зависимости от использования процессора, похоже на энергопотребление, представленное на рис. 1.

EDRP – Energy and Deadline aware Resource Provisioning [7]. Авторы сосредоточились на проблеме минимизации затрат на облачные системы, повышая эффективность использования энергии, но гарантируя сроки выполнения пользовательских задач, определенных в соглашениях по качеству обслуживания (SLA). Они принимают во внимание два типа задач, независимые пакетные задачи и задачи с зависимостями.

Их модель расчета потребления электроэнергии в момент времени t включает статическое $P^x_{static}(t)$ и динамическое $P^x_{dynamic}(t)$ энергопотребление. Обе характеристики рассчитываются на основе процента загрузки процессора $Util_x(t)$, в котором учитываются только параметры используемой виртуальной машины $Q^x(t)$.

Авторы не делают различия между виртуальными машинами, на которых работают задачи, и виртуальными машинами, находящимися в ждущем режиме, поскольку фоновая деятельность процессора необходима даже в режиме простоя.

$P^x_{static}(t)$ – это константа больше нуля при $Util_x(t) > 0$ и 0 в противном случае.

Отношение между $P^x_{dynamic}(t)$ и $Util_x(t)$ является гораздо более сложным. Серверы имеют оптимальный уровень загрузки с точки зрения производительности на ватт, который определяется как Opt_x. Общепризнано, что для современных серверов $Opt_x \approx 0.7$ и увеличение энергопотребления за пределами этой операционной точки более значительно, чем при $Util_x(t) < Opt_x$.

Несмотря на идентичные условия использования, энергоэффективность различных серверов может отличаться. Это отражается в коэффициентах αx и βx, представляющих увеличение энергопотребления Dx при $Util_x(t) < Opt_x$ и $Util_x(t) \geq Opt_x$ соответственно. $P^x_{dynamic}(t)$ рассчитывается как:

$$\begin{cases} \mathbf{Util_x(t) * \alpha_x} & \mathbf{if\ (Util_x(t) < Opt_x)} \\ \mathbf{Opt_x * \alpha_x + (Util_x(t) - Opt_x)^2 * \beta_x} & \mathbf{if\ (Util_x(t) \geq Opt_x)} \end{cases}$$

Предположим, что L_{max}. – это максимальная длина расписания всех приложений. Общее потребление энергии COSP – это сумма энергопотребления всех серверов по всему интервалу времени:

$$\mathbf{COSP} = \sum_{x=1}^{M} \left(\sum_{t=1}^{L_{max}} \left(\mathbf{P^x_{static}(t) + P^x_{dynamic}(t)} \right) \right).$$

Полученный процент потребляемой мощности показан на рис. 3.

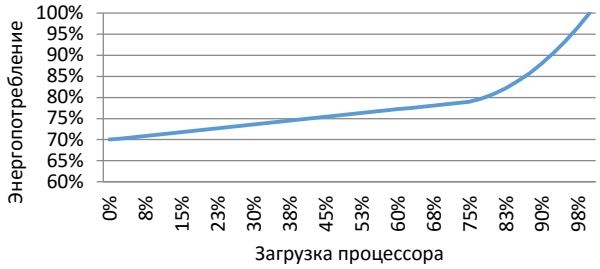

Рис. 3. Нелинейное энергопотребление в зависимости от загрузки процессора (%).

BFDP – Best Fit Decreasing Power [8]. Авторы предлагают методику расчета энергопотребления на основе полиномиальной регрессии Lasso, и алгоритма планирования ресурсов BFDP. Они направлены на повышение энергоэффективности без ухудшения качества обслуживания с учетом четырех типов задач, CPU-интенсивных, интенсивно работающих с памятью, сетью и системами ввода-вывода. Авторы вводят механизм порогов загрузки в BFDP, чтобы решить проблему чрезмерной консолидации. Результаты показали, что этот алгоритм создает меньше нарушений SLA. Нелинейная модель энергии записывается как функция:

$$\boldsymbol{y_i = \beta_0 + \sum_{j=1}^{m} \beta_j \phi_j(x_i) + \varepsilon_i,}$$

где $\phi_j(x_i)$ является ядром выражения y_i; x_i представляет загрузку процессора и памяти; β_i – это константа определяемая на основе модели обучения; ε_i – это константа. Рис. 4 показывает энергопотребление в зависимости от использования процессора и памяти.

PAHD – Power-aware Applications Hybrid Deployment [9]. Авторы рассматривают ресурсоемкие приложения и приложения, интенсивно использующие ввод/вывод для оптимизации ресурсов в виртуальных средах.

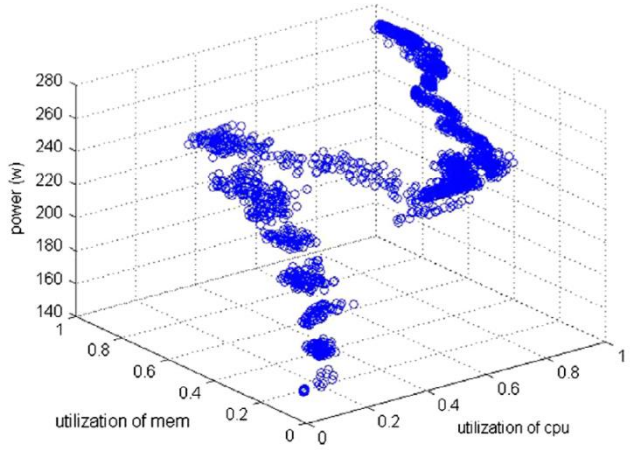

Рис. 4. Общее энергопотребление в зависимости от использования процессора и памяти [8]

Они используют кроссплатформенный гипервизор Xen для мониторинга виртуальных машин. Авторы оценивают эффективность энергосбережения в диапазоне 2%-12% для различных алгоритмов распределения ресурсов, Они приходят к выводу, что если для ресурсоемких приложений выделяется в два раза больше ресурсов, по сравнению с приложениями интенсивно использующих ввод/вывод, то достигается улучшение энергоэффективности.
Таблица 1 суммирует основные характеристики приведенных алгоритмов и критерии, используемые для оценки их производительности.

Таблица 1. Основные характеристики рассматриваемых алгоритмов.

	Применение			Характеристики												Критерии			Лит
	Центры данных	Облака	Гибридные облака	Централизованный	Децентрализованный	Динамическая загрузка	Статическая загрузка	Стат. время выполнения	Дин. время выполнения	Качество обслуживания	Миграция	Статическая	Динамическая	ЦПУ интенсивные	В/В интенсивные	Загрузка	Энергия	Срок выполнения	
EMVM		●		●		●			●	●	●		●	●		●	●	●	[1]
HSFL		●		●		●				●	●		●	●		●	●		[2]
AETC		●		●			●	●		●	●			●		●	●		[3]
CTES		●			●	●		●		●			●	●		●	●	●	[4]
DVMA		●			●					●	●			●		●	●		[5]
EDRP		●		●			●	●		●	●	●		●		●	●	●	[6]
BFDP		●		●				●		●	●		●	●	●	●	●		[7]

PAHD	•								•				•	•	•	•		[8]

3. Общая формулировка проблемы оптимизации

Мы рассматриваем m однородных серверов, описываемых множеством {s, mem, band, eff }, где s – это скорость исполнения инструкций (MIPS), mem – объем памяти (МБ), band - доступная полоса пропускания (Мбит/с), и eff – эффективность использования энергии (MIPS на ватт). Мы исходим из того, что компьютеры имеют достаточно ресурсов для выполнения задач. Основная цель предлагаемых стратегий заключается в минимизации общего энергопотребления E.

3.1 Модель задачи

Мы рассматриваем n независимых задач $J_1, J_2, \dots, J_n$, где каждая задача J_j описывается множеством $J_j = (r_j, p_j, type_j)$. $r_j \geq 0$ это время запуска задачи, p_j – время решения задачи. Время запуска задачи r_j неизвестно до момента ее запуска. $type_j$ характеризует тип задачи.

3.2 Модель энергопотребления

Мы используем нелинейную гибридную модель потребления энергии, предложенную в [25]. Мы считаем, что выполнение задач различных типов на одном и том же процессоре по-разному влияет на энергопотребление в связи с использованием различных аппаратных средств.

Наша модель учитывает энергопотребление индивидуальных задач и их комбинаций. Из-за разнообразия задач и их комбинаций, мы предлагаем использовать суммарную загрузку процессора задачами каждого типа (общей загрузкой, которая обеспечивается каждым типом задач). В этой статье мы рассматриваем 2 типа задач (тип А и тип Б).

Рис.5 описывает нормализованное энергопотребление при решении этих задач в зависимости от загрузки процессора. Назначение двух задач разного типа на одном процессоре может вызвать сниженное энергопотребление, меньшее, чем сумма энергопотребления при решении каждой задачи по отдельности.

Назначение же нескольких задач одного типа на один процессор может негативно влиять на производительность, создавая узкие места в процессоре, диске или сети, что может привести к дополнительной деградации производительности системы и увеличению энергопотребления.

Энергопотребление процессора в момент времени t состоит из двух частей – энергопотребление в режиме ожидания, когда процессор включен, но не используется $e_{idle_i}^{proc}$, и энергопотребление, когда процессор используется $e_{used_i}^{proc}(t)$:

$$\boldsymbol{e_i^{proc}(t) = o_i(t) * \left(e_{idle_i}^{proc} + e_{used_i}^{proc}(t) * U_i(t)^r\right)}$$

где $o_i(t) = 1$, если процессор включен, и $o_i(t) = 0$, в противном случае. $U_i(t)$ загрузка процессора в момент времени t. r – коэффициент, предложенный в [15], чтобы учесть нелинейные свойства энергопотребления.

$$\mathbf{e_{used_i}^{proc}(t)} = \left(\left(e_{max_i}^{proc} - e_{idle_i}^{proc} \right) * \boldsymbol{\beta}(\boldsymbol{\alpha_A(t)}) \right)$$

где $e_{max_i}^{proc}$ – это максимальная потребляемая мощность, когда процессор полностью загружен. $\beta(\alpha_A(t))$ – это коэффициент, который показывает приращение энергопотребления, когда процессор работает с различными типами приложений. Концентрация задач типа A в момент времени t определяется как $\alpha_A(t)$.

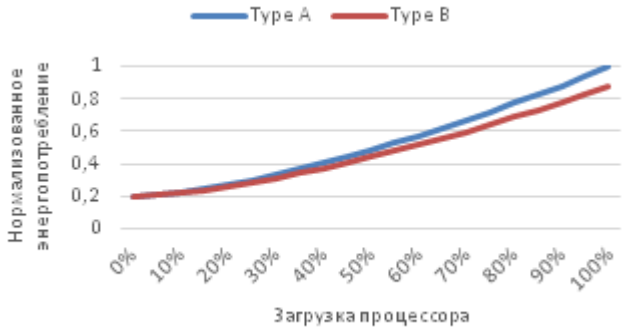

Рис. 5. Нормализованное энергопотребление задач типа A и B в зависимости от загрузки процессора (%).

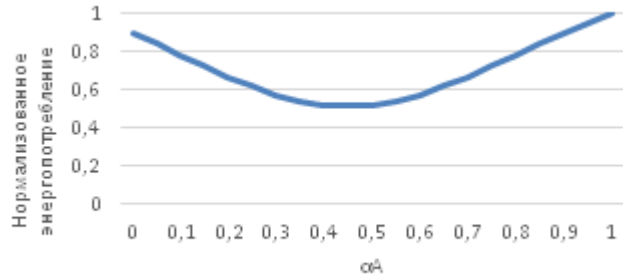

Рис. 6. Нормализованное энергопотребление в зависимости от пропорции задач типа A.

Рис. 6 показывает долю энергопотребления, когда процессор обрабатывает задачи типа А и Б.

Общая мощность, потребляемая системой подсчитывается как интеграл энергопотребления за время решения всех задач.

$$E^{op} = \int_{t=1}^{C_{max}} E^{op}(t)dt, \text{ with } E^{op}(t) = \sum_{i=1}^{m} e_i^{proc}(t)$$

Мы используем $e_{idle}{}_i^{proc} = 0.2 * e_{max}{}_i^{proc}$ и устанавливаем $\beta(\alpha_A) = 1$ для $\alpha_A = 1$ (все задачи это задачи типа А) и $\beta(\alpha_A) = 0.9$ для $\alpha_A = 0$ (все задачи это задачи типа В).

Рис. 7 показывает нормализованное энергопотребление в зависимости от загрузки процессора и доли (%) задач типа А, когда процессор обрабатывает задачи двух типов.

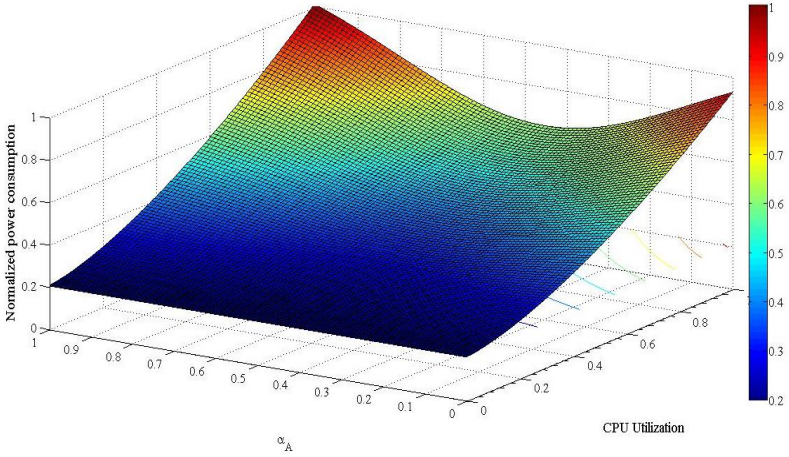

Рис. 7. Нормализованное энергопотребление задач типа А и В в зависимости от загрузки процессора (%) и пропорции задач типа А (%).

Мы видим, что когда загрузка процессора маленькая, энергопотребление низкое (голубая зона). В этом случае любая комбинация задач незначительно влияет на общее энергопотребление, и нижняя голубая зона имеет плоскую поверхность. Левая и правая зоны поверхности отражают преобладание задач типа А и задач типа Б, соответственно. В обоих случаях мы видим, что когда загрузка увеличивается до самого высокого уровня, энергопотребление также увеличивается до самого высокого уровня (красная зона). С другой стороны, если загрузка процессора сбалансирована между двумя типами задач, даже если загрузка увеличивается до самого высокого уровня (80-100%), энергопотребление остается в зеленой зоне и не достигает самых больших значений.

4. Алгоритмы планирования

В этом разделе мы описываем наш подход к планированию задач и энергосберегающие алгоритмы.

Мы используем базовый двухуровневый подход к планированию [12, 13, 14, 15]. На верхнем уровне система имеет общую информацию о задачах и доступных ресурсах, и назначает задачи на машины используя определенный критерий. Локальное распределение ресурсов для выполнения задач происходит на нижнем уровне.

Таблица 2. Стратегии назначения ресурсов.

Тип	Стратегия	Описание
Knowledge Free	Rand	Назначает задачу j на подходящую машину случайно используя равномерное распределение в диапазоне $[1..m]$.
	FFit (First Fit)	Назначает задачу j на первую машину, которая может выполнить ее.
	RR (Round Robin)	Назначает задачу j на машину, которая может выполнить ее используя стратегию Round Robin
	Min_L (Min load)	Назначает задачу j на машину с минимальной загрузкой в момент времени r_j: $min_{i=1..m}\{n_i\}$,
Energy aware	Min_Te (Min-Total_energy)	Назначает задачу j на машину с минимальным общим энерго-потреблением на момент времени r_j: $min_{i=1..m}\left(\sum_{t=1}^{r_j} e_i^{proc}(t)\right)$
	Min_e (Min-energy)	Назначает задачу j на машину с минимальным энергопотреблением в момент времени r_j: $min_{i=1..m}\left(e_i^{proc}(r_j)\right)$
Utilization aware	Min_u (Min-utilization)	Назначает задачу j на машину с минимальным уровнем загрузки (utilization) в момент времени r_j $min_{i=1..m}\left(u_i^{proc}\right)$
	Max_u (Max-utilization)	Назначает задачу j на машину с максимальной загрузкой (utilization) в момент времени r_j $max_{i=1..m}\left(u_i^{proc}\right)$
	Min_ujt (Min-util_job_type)	Назначает задачу j на машину с минимальной загрузкой (utilization) задачами того же типа в момент времени r_j
	Min_c (Min-concentration)	Назначает задачу j на машину с минимальной концентрацией задач того же типа в момент времени r_j

Процесс принятия решений о назначении задач на ресурсы основан на разных критериях. В этой статье мы изучаем 10 стратегий: Rand, FFit (First Fit), RR (Round Robin), Min_L (Min load), Min_Te (Min Total_energy), Min_e (Min energy), Min_u (Min utilization), Max_u (Max utilization), Min_ujt (Min utilization of job type), and Min_c (Min-concentration). (Таб. 2)

Мы разделяем их на три группы в зависимости от типа и количества информации, используемой для принятия решения: (1) knowledge-free, без использования информации о задачах и ресурсах [16, 17, 18]; (2) energy-aware, с информацией об энергопотреблении; (3) uilization-aware, с информацией о загрузке процессора.

5. Параметры экспериментов

В этом разделе мы описываем параметры экспериментов, включая совокупность задач, сценарии и методологию, используемую для анализа. Все эксперименты выполнены с использованием CloudSim: системы моделирования облачных вычислений, инфраструктуры и сервисов. Это стандартная программа, используемая для изучения распределения задач в облачных вычислениях. Мы расширили CloudSim для того, что включить наши алгоритмы, используя Java (JDK 7u51).

5.1 Загрузка задач

Анализ производительности алгоритмов планирования требует тщательного проведения экспериментов. Важным элементом является использование реальных задач. Производительность наших алгоритмов оценена на реальных задачах, взятых из архива задач высокопроизводительных вычислительных систем Parallel Workloads Archive [21], и вычислительных гридов Grid Workload Archive [22].

Загрузка системы (workloads) сформирована из 9 систем (traces): DAS2-University of Amsterdam, DAS2–Delft University of Technology, DAS2–Utrecht University, DAS2–Leiden University, KHT, DAS2–Vrije University Amsterdam, HPC2N, CTC, and LANL. Детальная информация о характеристике этих систем и задачах описана в [21, 22].

Характеристики задач, такие как, их распределение по дням, неделям, часовым поясам, может повлиять на правильную оценку эффективности алгоритмов. Таким образом, нам необходимо использовать нормализацию времени путем сдвига времени запуска задач на определенный интервал, чтобы обеспечить более реальные условия. Мы проводим нормализацию задач по часовым поясам и фильтрацию задач с ошибками. Мы помещаем все задачи в тот же самый часовой пояс, так что загрузка всех машин начинается в тот же самый день недели и время суток. Отметим, что выравнивание связано с местным временем, следовательно, поддерживается разница, соответствующая исходному часовому поясу. Применены несколько фильтров, чтобы удалить задачи с некорректной информацией, например, с отрицательным временем запуска, временем выполнения, количеством требуемых процессоров, пользовательским индексом и т.д. Мы также добавляем два поля в файлы загрузки для определения типа задач и загрузки процессоров.

Мы рассматриваем динамическую проблему планирования, в которой решения принимаются без полной информации о проблеме и задачах. Задачи прибывают одна за другой, время обработки задач неизвестно до тех пор, пока задача не закончится.

Рис. 8 показывает распределение задач по неделям. Он изображает общее количество задач, количество задач типа A и типа B. Мы видим, что

отсутствует преобладание одного типа задач в регистрах загрузки. В некоторые недели обрабатывается больше задач типа А, в другие – типа Б. Чтобы получить правильные статистические данные, мы используем 30 недель.

Рис. А1, А2 и А3 в Приложении показывают гистограмму общего числа задач в час, среднее число задач в день, и в час, соответственно. Рис. А4 и А5 в Приложении демонстрируют распределение каждого вида задач в неделю и в час для более детального анализа.

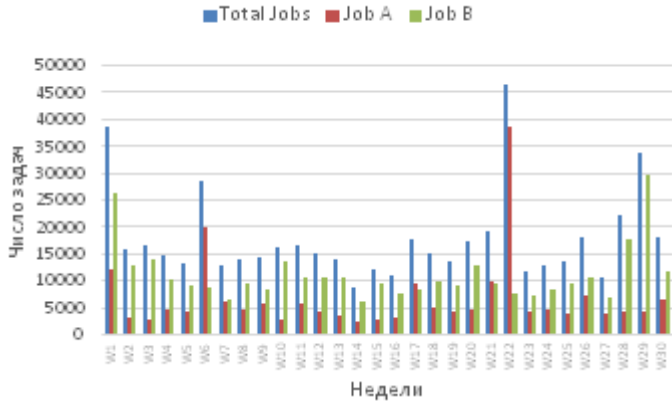

Рис. 8. Общее число задач, число задач типа A и типа B по неделям.

5.2 Сценарии

Следуя статье [11], которая описывает энергопотребление процессора Fujitsu PRIMERGY TX300 S7, мы используем $e_{max_i}^{proc} = 300$, $e_{idle_i}^{proc} = 0.2 * e_{max_i}^{proc}$ и устанавливаем нелинейное свойство энергопотребления $r = 1.5$.

6. Методология анализа

6.1 Деградация производительности

Чтобы обеспечить правильный выбор наилучшей стратегии, мы проводим анализ энергопотребления на основе методологии средней деградации, предложенной Tsafrir [24], и примененной для планирования задач в [12, 17, 20].

Прежде всего, мы оцениваем деградацию в производительности (относительную ошибку) каждой стратегии для каждой метрики. Эта деградация оценивается относительно наилучшего результата полученного всеми стратегиями для каждой метрики:

$$(\boldsymbol{\gamma - 1}) * \mathbf{100} \text{ with } \boldsymbol{\gamma} = \frac{\boldsymbol{strategy\ metric\ value}}{\boldsymbol{best\ found\ metric\ value}}$$

Затем мы усредняем эти значения и ранжируем стратегии. Наилучшая стратегия самой низкой средней деградации имеет ранг 1. Отметим, что мы пытаемся найти стратегию, которая работает эффективно в разных сценариях, таким образом, мы пытаемся найти компромисс, который учитывает все условия. Например, ранг стратегии может быть разным в разных сценариях, поэтому мы подсчитываем среднюю деградацию, чтобы оценить производительность стратегий и показать есть ли стратегии, которые доминируют над другими. Этот подход анализирует результаты на основе средних значений, однако, небольшая часть данных с большим разбросом может повлиять на средние значения. Для того, чтобы лучше интерпретировать данные, мы используем профили производительности наших стратегий.

6.2 Профиль производительности

Профиль производительности $\rho(\tau)$ это неубывающая, кусочно-постоянная функция, которая представляет вероятность того, что отношение γ находится в факторе τ от лучшего значения [23]. Функция $\rho(\tau)$ это накапливающая функция распределения. Стратегии с большой вероятностью $\rho(\tau)$ при малых τ будут предпочтительнее.

7. Экспериментальный анализ

В этом разделе мы приводим результаты экспериментального анализа предлагаемых стратегий. Во-первых, мы оцениваем их на основе методологии деградации производительности, описанной в разделе 6.1, и ранжируем их. Затем, мы проводим более детальный анализ, основанный на профилях производительностей стратегий.

7.1 Деградация энергопотребления

Рис. 9 показывает среднюю деградацию энергопотребления эвристик за неделю. Небольшой процент деградации показывает, что стратегия получает результаты, которые близки к наилучшем результатам, полученными всеми стратегиями. Таким образом, маленькая деградация демонстрирует лучшие результаты.

Мы наблюдаем, что деградации сильно различаются в зависимости от недели и стратегии. Однако, Max_u показывает худшее поведение среди всех, а Min_c и Min_ujt - лучшее. Последние две стратегии доминируют во всех тестах.

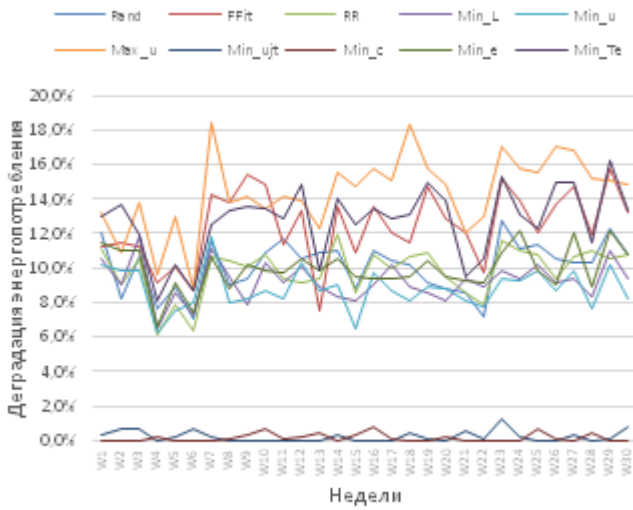

Рис. 9. Деградация энергопотребления по неделям.

Рис. 10 показывает среднюю деградацию энергопотребления по всем тестам. Стратегии Max_u, Min_Te и FFit показывают наихудшие значения деградации и ранжируются 10, 9, 8, соответственно. Min_c и Min_ujt являются наилучшими стратегиями с наименьшей деградацией и имеют ранги 1 и 2, соответственно.

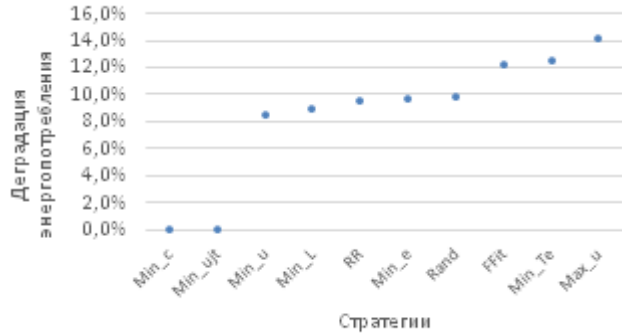

Рис. 10. Средняя деградация энергопотребления.

7.2. Профиль производительности

Как упоминалось в разделе 6, использование средних значений, может привести к неправильным выводам, так как небольшая часть результатов с большими отклонениями может существенно изменить среднее значение. Для того, чтобы проанализировать результаты более детально, мы учитываем профили производительности наших стратегий.

Рис. 11 показывает профили энергопотребления наших 10 стратегий в интервале τ=[0,...,0.2]. Мы видим больший разброс в деградации энергопотребления на значительном количестве тестов.

Min_c имеет наивысший ранг и самую высокую вероятность быть наилучшей стратегией. Если мы хотим получить результатах не хуже чем 1% (в факторе τ=0,01) от лучших найденных, вероятность того, что эта стратегия наилучшая для данной проблемы близка к 1. Min_ujt имеет второй ранг, вероятность того, что это лучшая стратегия с фактором 0.01 равна 0,96.

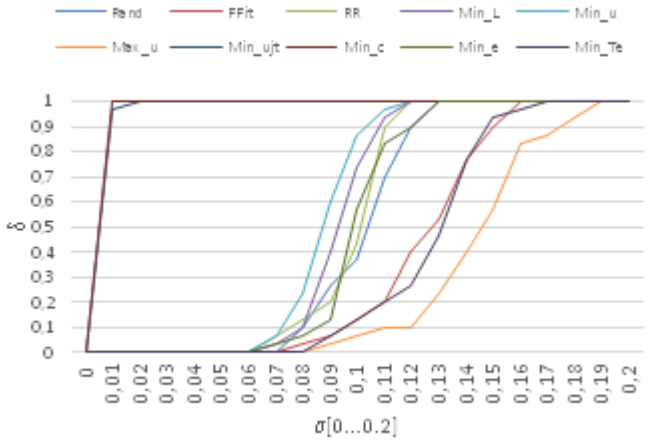

Рис. 11. Профиль деградации энергопотребления 10 стратегий.

8. Заключение

В этой статье мы предлагаем новый подход к распределению ресурсов с учетом характеристик задач. Основная идея нашего подхода основана на том факте, что различные задачи требуют различных ресурсов. Они могут быть; вычислительно-интенсивные (computing bound), интенсивно использующие процессор; интенсивно использующих ввод/вывод (I/O bound), требующие высокой пропускной способности; memory bound, disk bound, и тд.

Когда задачи одного типа назначаются на один и тот же ресурс, они могут создать узкие места и конфликты по использованию ресурсов в процессоре, в

памяти, на диске или в сети. Это может привести к деградации производительности и увеличению энергопотребления.

Современные планировщики не используют информацию о типах задач. Мы предлагаем использовать эту информацию контролируемым способом, для того, чтобы использовать ресурсы более эффективно и снизить энергопотребление.

Мы спроектировали и проанализировали новые алгоритмы планирования сосредоточившись на стратегиях назначения задач на вычислительные, которые принимают во внимание как информацию о динамическом состоянии ресурсов так и о типе задач.

Основные результаты можно сформулировать следующим образом.

(a) Мы сформулировали проблему распределения задач с различными характеристиками для оптимизации энергопотребления;

(b) Мы предложили и провели всестороннее исследование производительности 10 стратегий используя систему моделирования облачных вычислений CloudSim на реальных данных в различных сценариях.

(c) Для того, чтобы обеспечить выбор наилучшей стратегии, мы применили анализ на основе методологии оценки деградации производительности каждой стратегии и использовали их профили производительности.

(d) Мы обнаружили, что информация о загрузке процессора без информации о типе задач не помогает существенно улучшить его энергопотребление.

(e) Основываясь на этих результатах, мы показали, что, учитывая тип задач, мы можем оптимизировать их распределение, тем самым снизить энергопотребление и увеличить производительность, избегая создания узких мест и конфликтов по ресурсам.

(f) Результаты детального моделирования, представленные в статье, показывают, что с точки зрения минимизации энергопотребления, стратегия, которая балансирует концентрацию задач различных типов Min_c превосходит другие алгоритмы. Эта стратегия приводит к результатам, которые доминирую почти во всех тестах. Мы пришли к выводу, что стратегия стабильна в различных условиях. Она обеспечивает незначительную деградацию производительности при различных сценариях.

Тем не менее, необходимо дальнейшее изучение предлагаемого подхода для оценки потребления энергии. Учет нескольких типов задач и их концентраций на реальных вычислительных ресурсах обязателен для оценки фактической эффективности стратегий и предложенного метода.

Важно тщательное профилирование и изучение характеристик задач разных типов. Кроме того, необходимо анализировать типы приложений и

разработать методологию разделения их на категории. Это будет предметом будущей работы для лучшего понимания типов задач и их влияния на энергопотребление.

Приложение

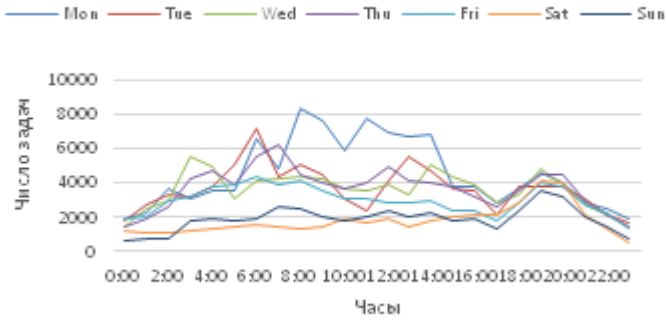

Рис. А1. Общее число задач в час в разные дни недели.

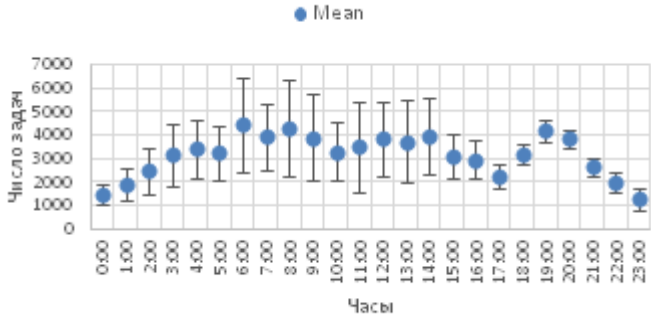

Рис. А2. Среднее число задач в день в течении часа.

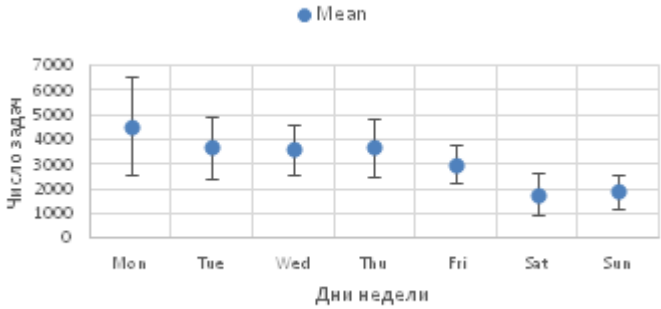

Рис. A3. Среднее число задач в день в течении недели

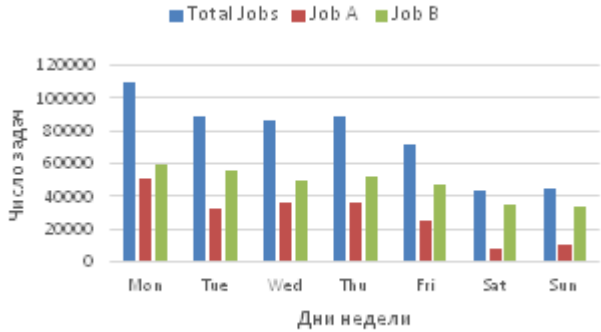

Рис. A4.Общее число задач типа A и B в день.

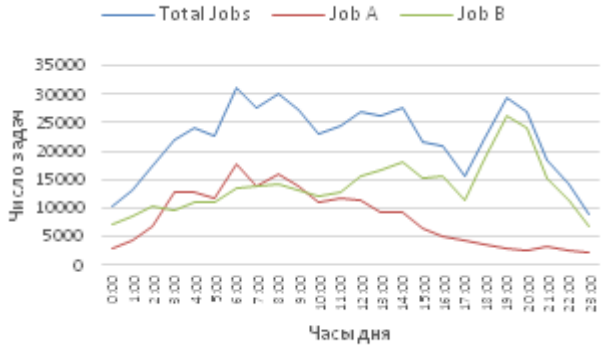

Рис. A5. Общее число задач типа A и B в час.

Литература

[1]. D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, A. Y. Zomaya, CA-DAG: Modeling Communication-Aware Applications for Scheduling in Cloud Computing, Journal of Grid Computing, 2015.

[2]. A. Beloglazov, J. Abawajy, and R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing, Future Gener. Comput. Syst., vol. 28, no. 5, pp. 755–768, May 2012.

[3]. J. Luo, X. Li, and M. Chen, Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers, Expert Syst. Appl., vol. 41, no. 13, pp. 5804–5816, Oct. 2014.

[4]. C.-H. Hsu, K. D. Slagter, S.-C. Chen, and Y.-C. Chung, Optimizing Energy Consumption with Task Consolidation in Clouds, Inf. Sci., vol. 258, pp. 452–462, Feb. 2014.

[5]. S. Hosseinimotlagh, F. Khunjush, and S. Hosseinimotlagh, A Cooperative Two-Tier Energy-Aware Scheduling for Real-Time Tasks in Computing Clouds, in Proceedings of the 2014 22Nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Washington, DC, USA, 2014, pp. 178–182.

[6]. X. Wang, X. Liu, L. Fan, and X. Jia, A Decentralized Virtual Machine Migration Approach of Data Centers for Cloud Computing, Math. Probl. Eng., vol. 2013, p. e878542, Aug. 2013.

[7]. Y. Gao, Y. Wang, S. K. Gupta, and M. Pedram, An Energy and Deadline Aware Resource Provisioning, Scheduling and Optimization Framework for Cloud Systems," in Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Piscataway, NJ, USA, 2013, pp. 31:1–31:10.

[8]. L. Luo, W. Wu, W. T. Tsai, D. Di, and F. Zhang, Simulation of power consumption of cloud data centers, Simul. Model. Pract. Theory, vol. 39, pp. 152–171, Dec. 2013.

[9]. Z. Liu, R. Ma, F. Zhou, Y. Yang, Z. Qi, and H. Guan, "Power-aware I/O-Intensive and CPU-Intensive applications hybrid deployment within virtualization environments," in 2010 IEEE International Conference on Progress in Informatics and Computing (PIC), 2010, vol. 1, pp. 509–513.

[10]. A. Lezama, A. Tchernykh, R. Yahyapour, Performance Evaluation of Infrastructure as a Service Clouds with SLA Constraints. Computación y Sistemas 17(3): 401–411 (2013).

[11]. S. B. Matthias Splieth, "Analyzing the Effect of Load Distribution Algorithms on Energy Consumption of Servers in Cloud Data Centers," 2015.

[12]. A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J. Pecero, S. Nesmachnow: Energy-Aware Online Scheduling: Ensuring Quality of Service for IaaS Clouds. International Conference on High Performance Computing & Simulation (HPCS 2014), pp 911–918, Bologna, Italy (2014).

[13]. A. Tchernykh, U. Schwiegelsohn, R. Yahyapour, N. Kuzjurin: Online Hierarchical Job Scheduling on Grids with Admissible Allocation, Journal of Scheduling 13(5):545–552 (2010)

[14]. A. Tchernykh, J. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin, S. Zhuk,: Two Level Job-Scheduling Strategies for a Computational Grid. In R. Wyrzykowski et al. (eds.) Parallel Processing and Applied Mathematics, 6th International Conference on Parallel Processing and Applied Mathematics. Poznan, Poland, 2005, LNCS 3911, pp. 774–781, Springer-Verlag (2006).

[15]. B. Dorronsoro, S. Nesmachnow, J. Taheri, A. Zomaya, E-G. Talbi, P. Bouvry: A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems. Sustainable Computing: Informatics and Systems 4:252–261 (2014).

[16]. A. Tchernykh, J. Pecero, A. Barrondo, E. Schaeffer: Adaptive Energy Efficient Scheduling in Peer-to-Peer Desktop Grids, Future Generation Computer Systems, 36:209–220 (2014).

[17]. J.M. Ramírez, A. Tchernykh, R. Yahyapour, U. Schwiegelshohn, A. Quezada, J. González, A. Hirales: Job Allocation Strategies with User Run Time Estimates for Online Scheduling in Hierarchical Grids. Journal of Grid Computing 9:95–116 (2011).

[18]. S. Iturriaga, S. Nesmachnow, B. Dorronsoro, P. Bouvry: Energy efficient scheduling in heterogeneous systems with a parallel multiobjective local search. Computing and Informatics 32(2):273–294 (2013)

[19]. U. Schwiegelshohn, A. Tchernykh: Online Scheduling for Cloud Computing and Different Service Levels, 26th Int. Parallel and Distributed Processing Symposium Los Alamitos, CA, pp. 1067–1074 (2012).

[20]. A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J. Pecero, S. Nesmachnow, A. Drozdov: Online Bi-Objective Scheduling for IaaS Clouds with Ensuring Quality of Service. Journal of Grid Computing, Springer-Verlag, DOI 10.1007/s10723-015-9340-0 (2015).

[21]. Parallel Workload Archive [Online, November 2014]. Available at http://www.cs.huji.ac.il/labs/parallel/ workload

[22]. Grid Workloads Archive [Online, November 2014]. Available at http://gwa.ewi.tudelft.nl

[23]. E. Zitzler: Evolutionary algorithms for multiobjective optimization: Methods and applications, PhD thesis, Swiss Federal Institute of Technology. Zurich (1999)

[24]. D. Tsafrir, Y. Etsion, D. Feitelson: Backfilling Using System-Generated Predictions Rather than User Runtime Estimates. *IEEE Transactions on Parallel and Distributed Systems* 18 (6), pp.789–803 (2007)

[25]. F. Armenta-Cano, A. Tchernykh, J. M. Cortés-Mendoza, R. Yahyapour, A. Drozdov, P. Bouvry, D. Kliazovich, A. Avetisyan: Heterogeneous Job Consolidation for Power Aware Scheduling with Quality of Service. Proceedings of the 1st Russian Conference on Supercomputing - Supercomputing Days 2015, Moscow, Russia, September 28-29, 2015. Editors V. Voevodin, S. Sobolev. Published on CEUR-WS: 22-Oct-2015, Vol-1482, p. 687-697. ONLINE: http://ceur-ws.org/Vol-1482/, URN: urn:nbn:de:0074-1482-7

Min_c: Heterogeneous Concentration Policy for Power Aware Scheduling[2]

[1]F. Armenta-Cano <armentac@cicese.edu.mx>
[1]A. Tchernykh <chernykh@cicese.mx>
[1]J. M. Cortés-Mendoza <jcortes@cicese.edu.mx>
[2]R. Yahyapour <ramin.yahyapour@gwdg.de>
[3]A. Yu. Drozdov <alexander.y.drozdov@gmail.com>
[4]P. Bouvry <Pascal.Bouvry@uni.lu>
[4]D. Kliazovich <Dzmitry.Kliazovich@uni.lu>
[5]A. Avetisyan <arut@ispras.ru>
[6]S. Nesmachnow <sergion@fing.edu.uy>
[1] CICESE Research Center, Carretera Ensenada-Tijuana No. 3918, Zona Playitas, Código Postal 22860, Apdo. Postal 360, Ensenada, B.C. México
[2]GWDG – University of Göttingen, Wilhelmsplatz 1, 37073 Göttingen, Göttingen, Germany
[3] Moscow Institute of Physics and Technology (State University) 9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia
[4]University of Luxembourg. Maison du Savoir, 2, Avenue de l'Université, L-4365 Esch-sur-Alzette, Luxembourg
[5]Institute for System Programming of the RAS, 25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia.
[6]Universidad de la República, Av. 18 de Julio 1968, 11200 Montevideo, Uruguay

Abstract. In this paper, we address power-aware online scheduling of jobs with resource contention. We propose an optimization model and present a new approach to resource allocation based on job concentration. We take into account different types of applications and heterogeneity of workloads that could include CPU-intensive, disk-intensive, I/O-intensive, memory-intensive, network-intensive and other applications. When jobs of one type are allocated to the same resource, they may create a bottleneck and resource contention either in CPU, memory, disk or network. It may result in system performance degradation and increasing energy consumption. The main objective is to minimize the total energy consumption of running heterogeneous workloads. We focus on energy characteristics of applications assuming that applications of different types contribute differently to the total power consumptions due to use different hardware. We propose a nonlinear hybrid model of energy consumption. Our model takes into account power consumption of individual jobs and their combinations. We propose heterogeneous job consolidation algorithms and validate them by conducting a performance evaluation study using the CloudSim toolkit under different scenarios and real data. We analyze several scheduling algorithms depending on the type and amount of information they require. We show that information about resources utilization without knowledge of jobs types does not help much to improve the total energy

[2] The work is partially supported by the Ministry of Education and Science of Russian Federation under contract No02.G25.31.0061 12/02/2013 (Government Regulation No 218 from 09/04/2010).

consumption. In the other hand, being aware of types of applications, intelligent allocation strategies can further improve energy consumption compared with traditional approaches.

Keywords: Energy efficiency, type of applications, resource contention, scheduling.

DOI: 10.15514/ISPRAS-2015-27(6)-23

For citation: Armenta-Cano F., Tchernykh A., Cortés-Mendoza J. M., Yahyapour R., Drozdov A.Yu., Bouvry P., Kliazovich D., Avetisyan A., Nesmachnow S. Min_c: Heterogeneous Concentration Policy for Power Aware Scheduling. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 6, 2015, pp. 355-380 (in Russian). DOI: 10.15514/ISPRAS-2015-27(6)-23.

References

[1]. D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, A. Y. Zomaya, CA-DAG: Modeling Communication-Aware Applications for Scheduling in Cloud Computing, Journal of Grid Computing, 2015.

[2]. A. Beloglazov, J. Abawajy, and R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing, Future Gener. Comput. Syst., vol. 28, no. 5, pp. 755–768, May 2012.

[3]. J. Luo, X. Li, and M. Chen, Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers, Expert Syst. Appl., vol. 41, no. 13, pp. 5804–5816, Oct. 2014.

[4]. C.-H. Hsu, K. D. Slagter, S.-C. Chen, and Y.-C. Chung, Optimizing Energy Consumption with Task Consolidation in Clouds, Inf. Sci., vol. 258, pp. 452–462, Feb. 2014.

[5]. S. Hosseinimotlagh, F. Khunjush, and S. Hosseinimotlagh, A Cooperative Two-Tier Energy-Aware Scheduling for Real-Time Tasks in Computing Clouds, in Proceedings of the 2014 22Nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Washington, DC, USA, 2014, pp. 178–182.

[6]. X. Wang, X. Liu, L. Fan, and X. Jia, A Decentralized Virtual Machine Migration Approach of Data Centers for Cloud Computing, Math. Probl. Eng., vol. 2013, p. e878542, Aug. 2013.

[7]. Y. Gao, Y. Wang, S. K. Gupta, and M. Pedram, An Energy and Deadline Aware Resource Provisioning, Scheduling and Optimization Framework for Cloud Systems," in Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Piscataway, NJ, USA, 2013, pp. 31:1–31:10.

[8]. L. Luo, W. Wu, W. T. Tsai, D. Di, and F. Zhang, Simulation of power consumption of cloud data centers, Simul. Model. Pract. Theory, vol. 39, pp. 152–171, Dec. 2013.

[9]. Z. Liu, R. Ma, F. Zhou, Y. Yang, Z. Qi, and H. Guan, "Power-aware I/O-Intensive and CPU-Intensive applications hybrid deployment within virtualization environments," in 2010 IEEE International Conference on Progress in Informatics and Computing (PIC), 2010, vol. 1, pp. 509–513.

[10]. A. Lezama, A. Tchernykh, R. Yahyapour, Performance Evaluation of Infrastructure as a Service Clouds with SLA Constraints. Computación y Sistemas 17(3): 401–411 (2013).

[11]. S. B. Matthias Splieth, "Analyzing the Effect of Load Distribution Algorithms on Energy Consumption of Servers in Cloud Data Centers," 2015.

[12]. A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J. Pecero, S. Nesmachnow: Energy-Aware Online Scheduling: Ensuring Quality of Service for IaaS Clouds. International Conference on High Performance Computing & Simulation (HPCS 2014), pp 911–918, Bologna, Italy (2014).

[13]. A. Tchernykh, U. Schwiegelsohn, R. Yahyapour, N. Kuzjurin: Online Hierarchical Job Scheduling on Grids with Admissible Allocation, Journal of Scheduling 13(5):545–552 (2010)

[14]. A. Tchernykh, J. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin, S. Zhuk,: Two Level Job-Scheduling Strategies for a Computational Grid. In R. Wyrzykowski et al. (eds.) Parallel Processing and Applied Mathematics, 6th International Conference on Parallel Processing and Applied Mathematics. Poznan, Poland, 2005, LNCS 3911, pp. 774–781, Springer-Verlag (2006).

[15]. B. Dorronsoro, S. Nesmachnow, J. Taheri, A. Zomaya, E-G. Talbi, P. Bouvry: A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems. Sustainable Computing: Informatics and Systems 4:252–261 (2014).

[16]. A. Tchernykh, J. Pecero, A. Barrondo, E. Schaeffer: Adaptive Energy Efficient Scheduling in Peer-to-Peer Desktop Grids, Future Generation Computer Systems, 36:209–220 (2014).

[17]. J.M. Ramírez, A. Tchernykh, R. Yahyapour, U. Schwiegelshohn, A. Quezada, J. González, A. Hirales: Job Allocation Strategies with User Run Time Estimates for Online Scheduling in Hierarchical Grids. Journal of Grid Computing 9:95–116 (2011).

[18]. S. Iturriaga, S. Nesmachnow, B. Dorronsoro, P. Bouvry: Energy efficient scheduling in heterogeneous systems with a parallel multiobjective local search. Computing and Informatics 32(2):273–294 (2013)

[19]. U. Schwiegelshohn, A. Tchernykh: Online Scheduling for Cloud Computing and Different Service Levels, 26th Int. Parallel and Distributed Processing Symposium Los Alamitos, CA, pp. 1067–1074 (2012).

[20]. A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J. Pecero, S. Nesmachnow, A. Drozdov: Online Bi-Objective Scheduling for IaaS Clouds with Ensuring Quality of Service. Journal of Grid Computing, Springer-Verlag, DOI 10.1007/s10723-015-9340-0 (2015).

[21]. Parallel Workload Archive [Online, November 2014]. Available at http://www.cs.huji.ac.il/labs/parallel/ workload

[22]. Grid Workloads Archive [Online, November 2014]. Available at http://gwa.ewi.tudelft.nl

[23]. E. Zitzler: Evolutionary algorithms for multiobjective optimization: Methods and applications, PhD thesis, Swiss Federal Institute of Technology. Zurich (1999)

[24]. D. Tsafrir, Y. Etsion, D. Feitelson: Backfilling Using System-Generated Predictions Rather than User Runtime Estimates. *IEEE Transactions on Parallel and Distributed Systems* 18 (6), pp.789–803 (2007)

[25]. F. Armenta-Cano, A. Tchernykh, J. M. Cortés-Mendoza, R. Yahyapour, A. Drozdov, P. Bouvry, D. Kliazovich, A. Avetisyan: Heterogeneous Job Consolidation for Power Aware Scheduling with Quality of Service. Proceedings of the 1st Russian Conference on Supercomputing - Supercomputing Days 2015, Moscow, Russia, September 28-29, 2015. Editors V. Voevodin, S. Sobolev. Published on CEUR-WS: 22-Oct-2015, Vol-1482, p. 687-697. ONLINE: http://ceur-ws.org/Vol-1482/, URN: urn:nbn:de:0074-1482-7