УДК 004.75

СИНТЕЗ ЭФФЕКТИВНЫХ БРОКЕРСКИХ РЕШЕНИЙ ПО ТЕХНИЧЕСКОЙ РЕАЛИЗАЦИИ АКТОРНЫХ ВЗАИМОДЕЙСТВИЙ ОБЛАЧНОЙ СРЕДЫ

Д.Ю. Воронин, М.С. Лисов, Ю.В. Сосновский*

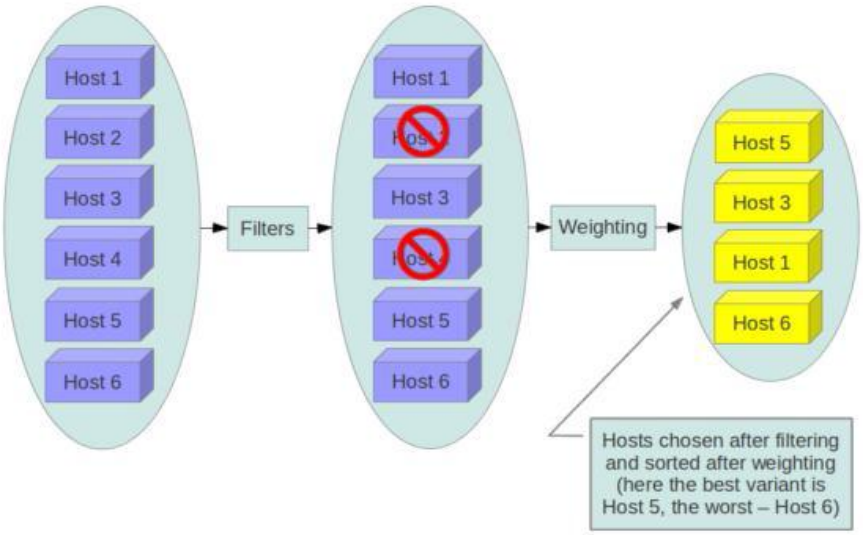
Севастопольский государственный университет
г. Севастополь, ул. Университетская, 33
E-mail: dima@voronins.com
* Крымский федеральный университет
г. Симферополь, пр. академика Вернадского, 4

При решении проблем критического компьютинга в облачных средах возникает острая необходимость в синтезе адаптивных интеллектуальных систем поддержки принятия решений, позволяющих Брокеру выбирать эффективную стратегию технической реализации акторных взаимодействий. Реализованы развитые инструментальные средства, позволяющие осуществлять интерактивный процесс поддержки принятия брокерских решений в реальном масштабе времени.

Введение. При решении проблем эффективной организации вычислительного обслуживания систем мониторинга окружающей среды все попытки рассмотреть облачную вычислительную среду (ОВС) как обычную систему, состоящую из подсистем, ввиду слабой конструктивности приводят к весьма ограниченным результатам. Альтернативным вариантом описания ОВС является референтная модель, предложенная в [1]. В рамках статьи рассматриваются следующие базовые Акторы референтной модели: Потребитель, Провайдер и Брокер. Их функциональные особенности и основные сценарии взаимодействия приведены в [1]. При классическом подходе подсистемы наделены четко выраженными функциями, ориентированными на достижение целевой функции всей системы, и именно в связи с этим оценивается их полезность, необходимость, эффект, степень развития [2]. Для ОВС это не совсем так – каждый из базовых Акторов имеет свой относительный характер поведения, свои цели, а качество организации ОВС во многом определяется коллинеарностью этих целей: насколько они приведены к общему ресурсному источнику и ориентированы на производство обобщенного продукта деятельности ОВС. Таким образом, задача синтеза эффективных брокерских решений по технической реализации акторных взаимодействий облачной среды является актуальной.

Аналитический обзор. В настоящее время существует множество альтернативных платформ построения облачных сервисов, например, Cloudstack, vCloudDirector, Openstack, Eucalyptus и многие другие. Согласно [3–5], наиболее перспективной является бесплатная платформа OpenStack, представляющая собой комплекс проектов свободного программного обеспечения, который может быть использован для создания инфраструктурных облачных сервисов и облачных хранилищ [6]. В OpenStack функционирует планировщик nova-schedule [7] (модель представлена на рис. 1), который осуществляет выбор сервера в соответствии со стратегией технической реализации акторных взаимодействий облачной среды. Она может варьироваться в зависимости от следующих характеристик: коэффициент загрузки виртуальных машин, доступный объем оперативной памяти и других вычислительных ресурсов.

В [8] приведен эвристический алгоритм диспетчеризации, основанный на описании поведения колонии муравьёв. Каждый агент хранит в памяти список пройденных им узлов. Этот список называют списком запретов (tabu list), моделирующем «память» агента. То есть, выбирая узел для следующего шага, агент учитывает информацию о пройденных узлах и не рассматривает их в качестве допустимых для перехода. На каждом шаге tabu list пополняется новым узлом, а перед новой итерацией алгоритма – то есть перед тем, как проходит формирование соответствующего пути – происходит очистка памяти. Кроме списка запретов при выборе узла для перехода учитывается «привлекательность» ребер, которая зависит, во-первых, от расстояния между узлами,



Р и с. 1. Фильтрация и стоимостное ранжирование серверов в платформе OpenStack

а, во-вторых, от следов феромонов, оставленных на ребре прошедшими по нему ранее агентами. Естественно, что в отличие от весов ребер, которые являются константными, следы феромонов обновляются на каждой итерации алгоритма.

В [9] подробно описан генетический алгоритм (GeneticAlgorithm), который представляет собой вариант стохастического поиска, в котором особи-потомки формируются путем изменения или комбинирования двух и более родительских особей. Функционирование алгоритма начинается с генерирования множества сформированных случайным образом особей, называемого популяцией. Каждая особь описывается в виде строки символов и оценивается при помощи функции пригодности (fitness function), которая формируется в соответствии с характерными особенностями решаемой задачи, например, связанной с оптимизацией распределения ресурсов.

В [10] рассмотрен списочный алгоритм, подразумевающий эвристическое упорядочивание заданий. Запуск очередного задания из списка происходит по мере доступности необходимого количества ресурсов. Списочные алгоритмы не требуют предварительной информации о длительностях выполнения заданий.

В работе [11] описан алгоритм планирования GT-CSB, основанный на использовании модели сетевого планирования. Согласно [12], применение метода Роя частиц при диспетчеризации заданий демонстрирует весьма интересные результаты. Предполагается, что текущее состояние агента характеризуется координатами допустимых решений, а также вектором скорости перемещения. Оба параметра выбираются случайным образом на этапе инициализации. Кроме того, каждая частица хранит найденный локальный и глобальный оптимум задачи, причем обмен информацией между частицами происходит мгновенно. На каждой итерации алгоритма направление и длина вектора скорости каждой из частиц изменяются в соответствие со сведениями о найденных оптимумах. После вычисления направления вектора, частица перемещается в вычисляемую точку. В случае необходимости, оптимумы обновляются и цикл повторяется.

В [13] приведена информация об использовании POSEC метода в Cloud-среде. В POSEC можно выделить следующие процессы: 1) Упорядочение заданий; 2) Оценка продолжительности действий; 3) Резервирование времени; 4) Принятие решений по приоритетам и перепоручением для заданий; 5) Контроль учёта выполненного задания. В работе [14] среди современных методов планирования выполнения поступающих заданий выделяется алгоритм планирования групп задач (Gang scheduling), осуществляющий распределение ресурсов Облачного Брокера между группами задач. Информация о требуемых услугах

классифицируется, причем все услуги одной группы могут обрабатываться параллельно. Брокер выделяет каждой группе некоторый квант времени.

По результатам проведенного аналитического обзора можно сделать ряд выводов:

1. Существующие платформы развертывания систем облачного компьютинга имеют высокую степень детализации, что не всегда удобного для задач, требующих эффективных решений на системном уровне;

2. Методы диспетчеризации заданий в облачных средах динамично развиваются [14–20], таким образом, имеется необходимость в создании формализационного базиса, позволяющего сравнивать их эффективность;

3. Для систем поддержки принятия решений, позволяющих Брокеру выбирать эффективную стратегию технической реализации акторных взаимодействий, предъявляются существенные требования к оперативности выдаваемых рекомендаций.

На решение этих и ряда других задач ориентирована данная научная работа. Ее основной целью является развитие и совершенствование подходов к парированию априорной неопределенности при принятии управляющих решений в ОВС.

Формализационный базис. Информационное обеспечение ОВС может быть представлено кортежем (1), который является в некотором смысле определенным формализационным базисом, позволяющим перейти к ряду задач анализа и параметрического оценивания различных подходов к моделированию базовых акторных взаимодействий для систем критического компьютинга. Рассмотренная детализация кортежа не является исчерпывающей и может расширяться по мере постановки и формулировки дальнейших задач.

$$\text{ИС} = \left\langle \mathrm{S}_{\text{И}}, \mathrm{SV}_{\text{И}}, Ц_{\text{И}}, R_{\text{И}}, \mathrm{H}_{\text{И}}, П_{\text{И}}, \pi_{\text{И}}, \mathrm{T}_{\text{И}}, У_{\text{И}}, \mathrm{U}_{\text{И}}, \chi_{\text{И}}, O_{\text{И}} \right\rangle, \tag{1}$$

где $\mathrm{S}_{\text{И}}$ – множество идентификаторов базовых Акторов ОВС (с указанием их типа: Потребитель, Брокер, Провайдер); $\mathrm{SV}_{\text{И}}$ – матрица инцидентности, описывающая наличие взаимосвязей между базовыми Акторами ОВС); $Ц_{\text{И}}$ – вектор целевых функций ОВС; $R_{\text{И}}$ – вектор, описывающий доступные объемы ресурсов ОВС; $\mathrm{H}_{\text{И}}$ – вектор, описывающий объем продукта, выработанного ОВС (например, объем К-вычислений); $П_{\text{И}}$ – скаляр, описывающий эвентуальный потенциал ОВС; $\pi_{\text{И}}$ – скаляр, описывающий реализуемый потенциал ОВС; $\mathrm{T}_{\text{И}}$ – вектор, описывающий технологию переработки $R_{\text{И}}$ в $\mathrm{H}_{\text{И}}$; $У_{\text{И}}$ – множество управляемых параметров ОВС; $\mathrm{U}_{\text{И}}$ – множество управляющих воздействий (в том числе выбирающие вариант технологической настройки и организующие базовые акторные взаимодействия); $\chi_{\text{И}}$ – вектор случайных возмущений рассматриваемой ОВС; $O_{\text{И}}$ – вектор ограничений рассматриваемой ОВС (терминальные сроки вычислительного облуживания крупномасштабных научных задач и т.п.).

Графовая модель акторных взаимодействий. Предлагаемая графовая модель предназначена для реализации отображений, описанных выражением (2).

$$\begin{cases} M: \left\langle \mathrm{S}_{\text{И}}, \mathrm{SV}_{\text{И}}, \mathrm{U}_{\text{И}} \right\rangle \to \left\langle \mathrm{H}_{\text{И}}, \pi_{\text{И}}, \mathrm{U}_{\text{И}} \right\rangle \\ \gamma: \left\langle \mathrm{S}_{\text{И}}, \mathrm{SV}_{\text{И}}, \mathrm{U}_{\text{И}} \right\rangle \to \left(\mathrm{Q}(t_k), D(t_k), \mathrm{V}(t_k) \right) \\ \eta: \mathrm{V}(t_k) \to \mathrm{u}_{\mathrm{k}}; \\ \mu: \mathrm{u}_{\mathrm{k}} \to \rho_{\mathrm{S_i}}^{\mathrm{k}}; \\ \Phi: \rho_{\mathrm{S_i}}^{\mathrm{k}} \to \pi_{\text{И}}(t_k). \end{cases}, \tag{2}$$

где γ – функция, формирующая граф $G(t_k)$ на основе информации, содержащейся в кортеже $\langle \mathrm{S_И}, \mathrm{SV_И}, \mathrm{U_И} \rangle$;

$G(t_k) \overset{\Delta}{=} (\mathrm{Q}(t_k), D(t_k), \mathrm{V}(t_k))$ – граф, описывающий акторные взаимодействия, осуществляемые в момент времени t_k; $Q(t_k)$ – узлы графа $G(t_k)$ (поставлены в соответствие Акторам ОВС); $D(t_k)$ – дуги сетевого графа $G(t_k)$ (поставлены в соответствие связям между Акторами); $\mathrm{V}(t_k)$ – веса дуг сетевого графа $G(t_k)$ (описывают параметры базовых акторных взаимодействий). То есть, например, веса дуг, входящих в *i*-ую вершину графа $G(t_k)$, поставлены в соответствие объемам ресурса, получаемого *i*-ым Актором-акцептором в ходе акторных взаимодействий. Веса дуг, исходящих из *i*-ой вершины графа $G(t_k)$, описывают объемы продукта, отдаваемого *i*-ым Актором-донором в результате акторных взаимодействий.

$\mathrm{u_k} = \eta(\mathrm{V}(t_k))$;

$\mathrm{u_k}$ – вектор, описывающий управление акторными взаимодействиями, осуществляемыми в момент времени t_k;

η – функция, формирующая управление акторными взаимодействиями на основе информации о весах дуг графа $G(t_k)$;

$\rho_{\mathrm{S_i}}^{\mathrm{k}}$ – объем произведенного продукта *i*-ым Актором в *k*-ый момент времени;

$\rho_{\mathrm{S_i}}^{\mathrm{k}} = \mu(\mathrm{u_k})$;

μ – функция, формирующая продукт Актора в соответствии с текущим управлением $\mathrm{u_k}$;

$\pi_{\mathrm{И}}(t_k)$ – реализуемый потенциал ОВС, оцененный в момент времени t_k;

$$\pi_{\mathrm{И}}(t_k) = \Phi\left(\left(\rho_{\mathrm{S_1}}^{\mathrm{k}}, \rho_{\mathrm{S_2}}^{\mathrm{k}}, \ldots, \rho_{\mathrm{S}_{|\bar{\mathrm{S}}_{И}|}}^{\mathrm{k}}\right)\right)$$

Φ – функция, агрегирующая продукт отдельных Акторов в реализуемый потенциал ОВС.

На основании этой модели может быть поставлена и решена следующая оптимизационная задача.

Дано: $\langle \mathrm{S_И}, \mathrm{SV_И}, Ц_{\mathrm{И}}, R_{\mathrm{И}} \rangle$, $\langle П_{\mathrm{И}}, \mathrm{T_И}, У_{\mathrm{И}}, \chi_{\mathrm{И}}, O_{\mathrm{И}} \rangle$. Необходимо для $\forall \mathrm{k} = \{1, 2, \ldots, \mathrm{m}\}$ найти $\mathrm{u}_k = \arg \underset{\mathrm{u}_k \in U_Д^k}{\mathrm{extr}}\, Ц_{\mathrm{И}}$, где $U_Д^k$ – множество, описывающее допустимые управляющие воздействия в соответствии с ограничениями, заданными вектором $O_{\mathrm{И}}$. Одним из вариантов описания $Ц_{\mathrm{И}}$ является:

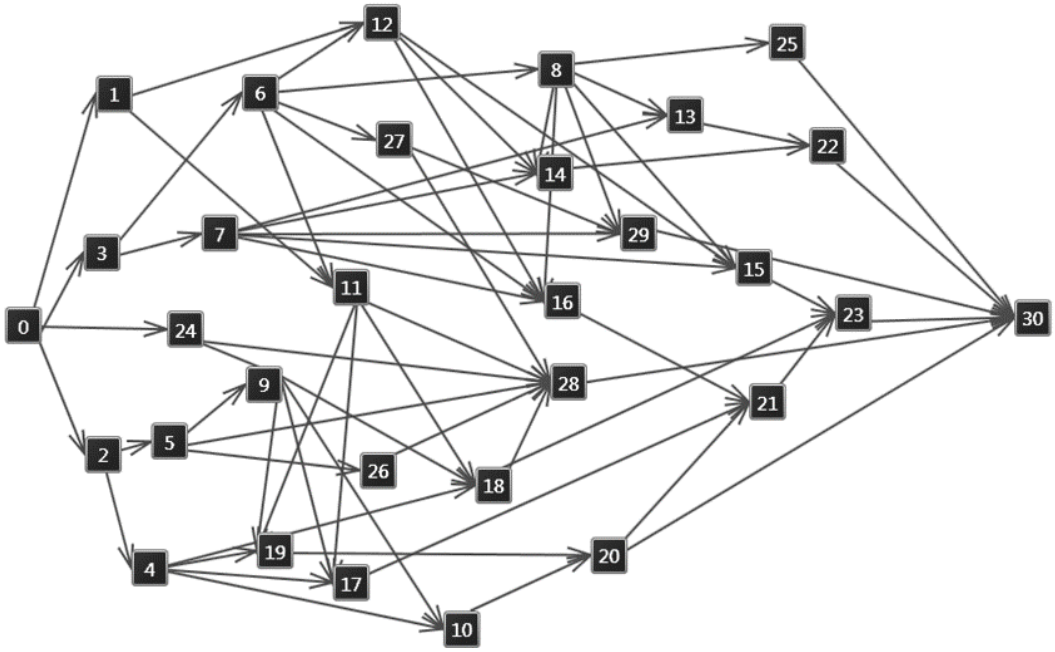
$$\frac{\pi_{\mathrm{И}}(t_k)}{П_{\mathrm{И}}} \to 1,\ \forall \mathrm{k} = \{1, 2, \ldots, \mathrm{m}\}.$$

Стратегии диспетчеризации. Задания, поступающие от Облачного Потребителя, состоят из множества отдельных работ, распределенных во времени. Между некоторыми из них может существовать отношение предшествования. Проект считается выполненным тогда и только тогда, когда выполнены все работы. Процесс их выполнения во времени удобнее всего представить в виде взвешенного сетевого графа (рис. 2). Появляется задача эффективного распределения ресурсов между работами проекта.

Можно выделить следующие наиболее эффективные стратегии брокера по обработке поступающих заданий: алгоритм «первого подходящего»; алгоритм циклического планирования; алгоритм поиска косяком рыб (FSS); метод роя частиц.

Метод «первый подходящий», является модификацией метода FCFS - провайдер предоставляется процессам в порядке их прихода в очередь готовности. First-Fit, допускает нарушение порядка очереди, позволяя низкоприоритетным заданиям занимать ресурсы, оставшиеся незагруженными. Это повышает эффективность их использования, но приводит к тому, что большая часть виртуальных машин занимается мелкими заданиями – возникает фрагментация ресурсов. По этой причине, даже если задание имеет самый высокий приоритет, необходимый ему объём ресурсов может никогда не образоваться и, следовательно, задание может никогда не стартовать – возникнет эффект «зависания» (starving).

Обычно такая проблема решается путем ограничения числа низкоприоритетных заданий, способных «обогнать»



Р и с. 2. Пример сетевого графа

более приоритетные задания. Это позволяет получить компромисс между предотвращением зависания заданий и эффективностью использования ресурсов. Данный метод имеет следующие преимущества:

- нет необходимости в информации о длительности выполнения заданий
- алгоритм прост в реализации и не требует больших вычислительных затрат.

Планирование по стратегии циклическое планирование (RR), предполагает, что каждая задача получает временной квант, в течение которого она может использовать ресурсы облака. Если завершение задачи не происходит по истечении кванта времени, то эта задача переводится в конец списка готовых к выполнению задач, а ресурсы облака предоставляются следующей задаче из списка. Очевидно, что для данного алгоритма планирования основной вопрос заключается в определении размера кванта времени, и следует ли делать его фиксированным или переменным. Очевидно, если квант времени выбирается слишком большим, то система RR фактически вырождается в FIFO, т.к. каждой задаче выделяется достаточно времени для завершения.

Если квант времени выбирается слишком малым, то контекстные переключения начинают играть доминирующую роль, что в итоге ухудшает характеристики системы и даже замедляет её работу.

В алгоритме FSS (Fish School Search) область поиска представляет собой аквариум (множество допустимых решений), в котором перемещаются рыбы (агенты). Как известно, в условиях поиска пищи (провайдера) рыбы плавают косяком, поэтому конечной целью является смещение всех агентов в область экстремума функции (оптимальный вариант распределения ресурсов).

Метод роя частиц предполагает, что текущее состояние частицы (агента) характеризуется координатами в пространстве решений (то есть, собственно, связанным с ними допустимым решением), а также вектором скорости перемещения. Оба параметра выбираются случайным образом на этапе инициализации. Кроме того, каждая частица хранит лучшее из найденных ей решений, а также лучшее из пройденных всеми частицами решений – имитируется мгновенный об-

мен информацией между частицами. На каждой итерации алгоритма направление и длина вектора скорости каждой из частиц изменяются в соответствие со сведениями о найденных оптимумах. После вычисления направления вектора, частица перемещается в вычисляемую точку. В случае необходимости, обновляются значения лучших точек для каждой частицы и для всех частиц в целом. После этого цикл повторяется.

Для моделирования взаимодействия акторов по выбранному сценарию воспользуемся многофункциональной системой имитационного моделирования Anylogic. Общий сценарий функционирования разрабатываемой системы, можно представить в следующем виде:

1. Формализация информационного описания, сетевой модели проекта.

1.1. Формирование структуры сетевого графа проекта

1.2. Указание всех необходимых параметров работ

1.3. Запись полученного формализованного описания в текстовый файл.

2. Переход к агентной парадигме.

2.1. Чтение из текстового файла параметров графа

2.2. Преобразование полученных данных в популяции агентов

2.3 Формирование необходимых статистических данных.

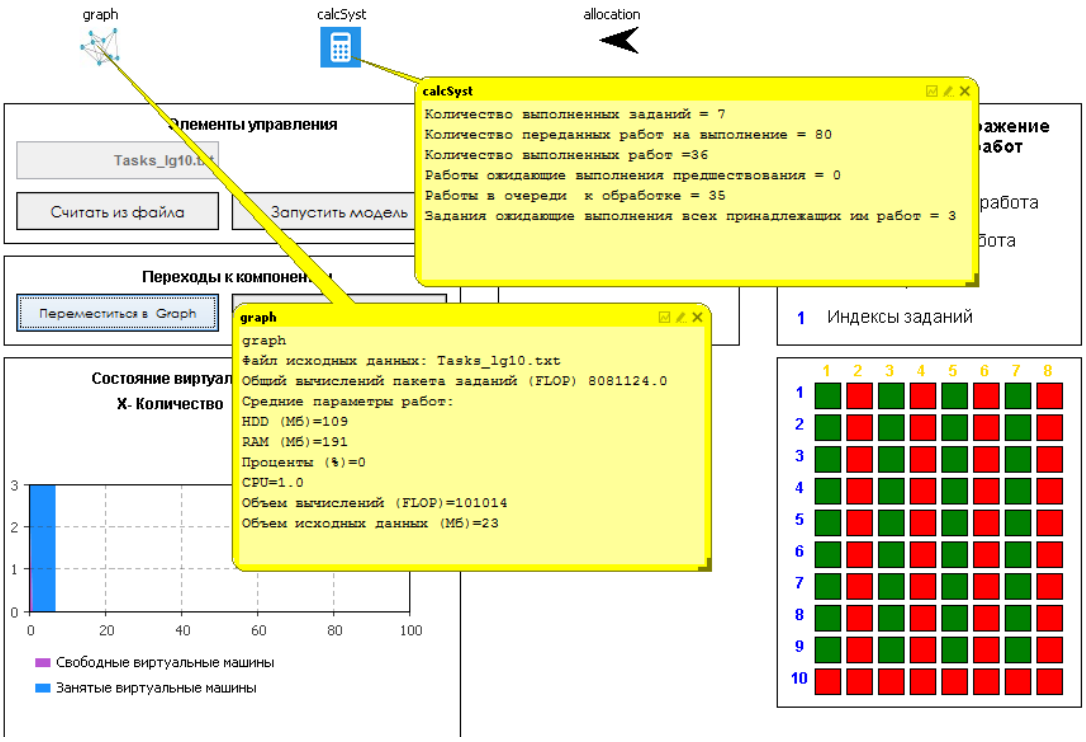
3. Моделирование обработки крупномасштабных задач облачными провайдерами.

3.1. Выбор Брокером диспетчерской стратегии при выборе Облачного Провайдера.

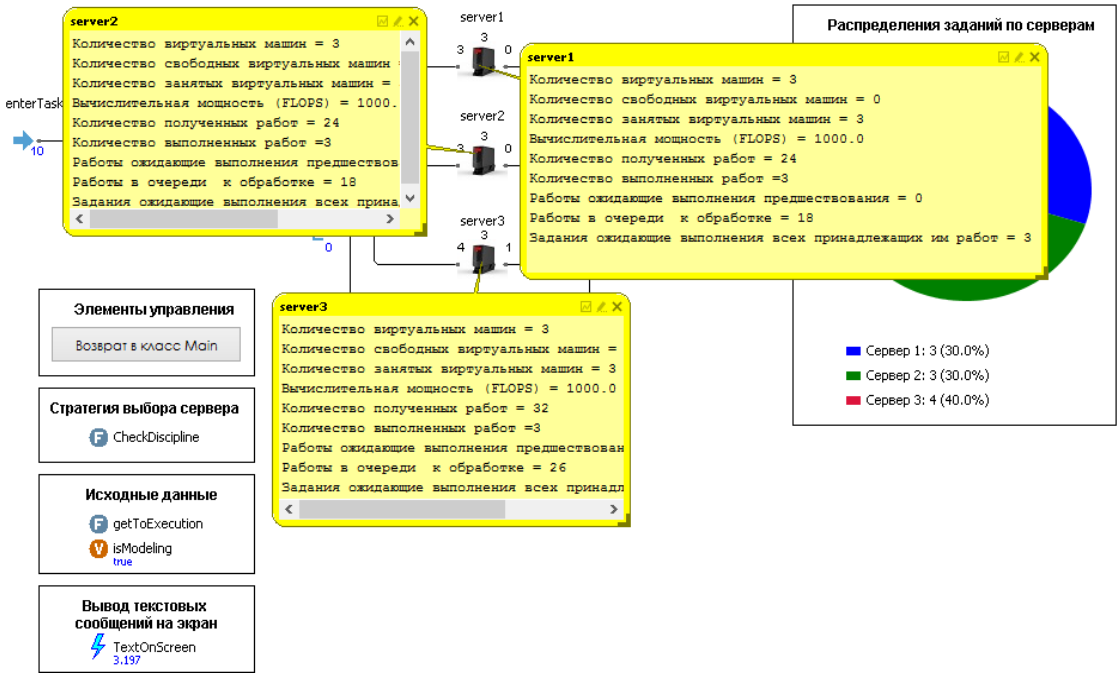
3.2. Моделирование обработки заданий Облачным Провайдером.

3.3. Формирование и анализ полученных результатов.

Пример диалогового окна вывода результатов моделирования реализованного стенда приведен на рис. 3. На пиктограммах изображена динамика выполнения работ в задании. Приведена статистическая информация, построены основные диаграммы. На рис. 4 представлен динамический процесс обновления статистических данных в классе Cloud CalcSyst.



Р и с. 3. Диалоговое окно реализованного стенда



Р и с. 4. Динамические характеристики Облачного Провайдера в процессе моделирования

Разработанный имитационный стенд ориентирован на предоставление ЛПР информации, необходимой для нахождения оптимального варианта распределения вычислительных ресурсов при использовании наиболее адекватного метода диспетчеризации.

Заключение. В работе рассмотрена задача формирования эффективной стратегии технической реализации акторных взаимодействий в соответствии со складывающейся информационной ситуации на решение крупномасштабных научных задач, используются различные математические абстракции, например, сетевые модели проектов. Однако, для автоматизации процедур поддержки принятия решений по управлению распределенной обработкой крупномасштабных вычислительных задач во многих случаях более эффективной является агентная парадигма. Рассмотренная в работе графовая модель в отличие от большинства известных подходов абстрагируется от конкретной инженерной реализации (что расширяет область применения), но в то же описывает процесс базового акторного взаимодействия на достаточном уровне детальей. Предложенное информационное обеспечение ОВС обладает важным системным свойством единства, что позволяет использовать его в качестве определенного формализационного базиса, позволяющего перейти к рассмотрению задач анализа и параметрического оценивания различных подходов к моделированию базовых акторных взаимодействий для критических ОВС. При иерархическом описании информационно-вычислительных работ, ориентированных лизации, приемлемом для принятия системных решений по управлению облачным вычислительным обслуживанием критических задач. Необходимо заметить, что приведенные в работе результаты не являются исчерпывающими. Перспективным направлением дальнейших исследований является разработка подходов, позволяющих детектировать аттракторы при формировании устойчивого управления ресурсами ОВС .

Работа выполнена в рамках задания по базовой части госзаказа темы № 3866.

СПИСОК ЛИТЕРАТУРЫ

1. *NIST* Cloud Computing Reference Architecture. Recommendations of the National Institute of Standards and Technology / F. Liu, J. Tong, J. Mao [et al.] .– [Электронный ресурс].– Электрон. текстовые дан. – Режим доступа: http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505, свободный (дата обращения: 24.11.2015).
2. *Прангишвили И.В.* Системный подход и общесистемные закономерности. – М: Синтег. – 2000. – 528 с.
3. *A Distributed* Behavioral Model / C. Reynolds, Flocks, Herds [et al.] // Computer Graphics. – 1987. – С. 307.
4. *Риз Д.* Облачные вычисления. – СПб.: БХВ-Петербург, 2011. – 288 с.
5. *Фингар П.* DOT.CLOUD. – М.: Аквамариновая книга, 2012. – 256 с.
6. *Comparison* of Workflow Scheduling Algorithms in Cloud Computing / K. Navjot, A. Singh, C. Singh // IJACSA. – 2011. – Vol. 2. – No. 10. – P. 10–18.
7. *OpenStack* [Электронный ресурс].– Электрон. текстовые дан. – Режим доступа: https://ru.wikipedia.org/wiki/OpenStack, свободный (дата обращения: 24.11.2015).
8. *Ant* Colony Algorithm Based Scheduling for Handling Software Project Delay / Wei Zhang, Yun Yang, Junchao Xiao [et al.] // IEEE Transactions on Software Engineering. – 2013. – 39 (1). – P. 1–17.
9. *Cormen T.* Introduction to Algorithms (Second Edition) // The MIT Press, NY. – 2009. – P. 237.
10. *Jackson D., Snell Q., Clement M.* Core algorithms of the Maui scheduler // Lecture Notes in Computer Science. – 2001. – Vol. 2221. – С. 763.
11. *Patel A., Darlington Y.* "A novel stochastic algorithm for scheduling QoS-constrained workflows in a web service-oriented grid," in Web Intelligence and International Agent Technology Workshops, 2006. WI-IAT 2006 Workshops // IEEE/WIC/ACM International Conference on. – 2006. – P. 437–442.
12. *Kosinska J., Kosinski J., Zielinski K.* The Concept of Application Clustering in Cloud Computing Environments // Valencia, 2010. – P. 139–145.
13. *Михайлов П.А., Радченко Г.И.* Методы моделирования и оценки производительности облачных систем // Вестник ЮУрГу. Серия: Вычислительная математика и информатика. – 2014. – № 3. – С. 109–123.
14. *An Algorithm* for Cost-Effectively Storing Scientific Datasets with Multiple Service Providers in the Cloud. / D. Yuan, X. Liu, L. Cui [et al.] // in 9th International Conference on e-Science e-Science, Atlanta, Georgia, USA. – 2010. – P. 285–292.
15. *Adaptive* control of virtualized resources in utility computing environments / P. Padala, K.G. Shin, X. Zhu [et al.] // SIGOPS Oper. Syst. Rev. – 2007. – Vol. 41. – P. 289–302.
16. *Boris M., Peter Van Roy*. From Mini-clouds to Cloud Computing. – Budapest, 2010. – P. 130–167.
17. *Security* prospects through cloud computing by adopting Multiple Clouds / M. Jensen, J. Schwenk, J.M. Bohli [et al.]. – Washington, DC. – 2011. – P. 565–572.
18. *An Algorithm* for Finding the Minimum Cost of Storing and Regenerating Datasets in Multiple Clouds / Dong Yuan, Lizhen Cui, Wenhao Li, Xiao Liu [et al.] // EEE TRANSACTIONS ON CLOUD COMPUTTING. – 2014. – P. – 20–37.
19. *Xiao J., Wang Q.* Value-based Multiple Software Projects Scheduling with Genetic Algorithm // International Conference on Software Process, Vancouver, Canada. – 2009. – P. 50–62.