УДК 004.942 DOI: 10.17587/it.24.464-474

С. В. Гаевой, канд. техн. наук, ст. преподаватель каф. ЭВМиС, e-mail: gaevserge@mail.ru;
В. М. А. Ахмед, аспирант каф. САПРиПК, e-mail: wesamalsofi@gmail.com;
С. А. Фоменков, д-р техн. наук, проф., проф. каф. САПРиПК, e-mail: saf@vstu.ru,
Волгоградский государственный технический университет, г. Волгоград, 400005

Использование имитационного моделирования для определения показателей обслуживания кластера

Важным способом анализа нагрузки вычислительного кластера является моделирование его работы (обслуживания приходящих заданий) с использованием модели входящей нагрузки (модели самих приходящих заданий). Нами уже построена сама имитационная модель кластера и определенный набор моделей входящей нагрузки. В данной работе рассмотрены особенности этой модели, ее программной реализации и процедура поиска конфигурации кластера на примере кластера UniLu-GAIA.

Ключевые слова: *нагрузки вычислительных систем, модели входящей нагрузки, немасштабируемые задачи, время выполнения заданий, имитационное моделирование, стохастическая аппроксимация, показатели обслуживания, вычислительные кластеры, программа для моделирования*

Введение

Проблема рационального выполнения параллельных и высокопроизводительных вычислений сейчас достаточно актуальна [1]. В частности, стоит вопрос оптимального балансирования нагрузки и подбора оптимальной производительности [2] вычислительных кластеров (ВК). Одним из возможных путей решения этой задачи является моделирование работы ВК — в том числе и имитационное [3]. Последнее требует построения математической модели ВК и поступающей на нее (т.е. входной) нагрузки [4].

В данной работе рассмотрим модель работы ВК, которая была построена нами ранее, особенности ее программной реализации и вопросы конфигурирования ВК, которые могут быть решены с помощью этих модели и программы.

1. Описание уже построенной модели ВК

Введем следующие определения из работы [2]. Число вычислительных машин, на которых исполняется задание, называется его шириной. Будем считать, что ширина задания определяется в момент его создания, что является довольно частым допущением [4—8] (немасштабируемые задания).

Длиной задания назовем время его выполнения на указанном в момент создания числе узлов (то есть на ширине). Вопрос изменения ширины не поднимается. Площадью задания назовем произведение длины на ширину. Очевидно, что площадь — это сложность задания. Она же представляет собой суммарное машинное время обслуживания. Этот параметр необходим для упрощения аппроксимации, так как многие математические модели в этом случае дают результат лучше. Далее эта особенность используется. Отметим, что авторы существующих публикаций по теме данной работы часто используют иную терминологию.

Реальный ВК построен из вычислительных машин, которые обслуживают поступающие задания [2, 3]. В базовой модели используется представление такого ВК в виде обслуживающего блока с единой неприоритетной, не ограниченной по размеру очередью. Это обеспечивает обслуживание всех входящих зада-

ний. При завершении исполнения очередного задания очередь просматривается вся от начала и до конца для выбора (извлечения) заданий на выполнение. При этом извлекаемых из нее заданий может быть несколько — ведь ушедшее из очереди (т.е. исполненное) задание может освободить не только один, но и несколько каналов (узлов) обслуживания. Первое обнаруженное задание, для выполнения которого имеется достаточное число свободных машин, ставится на исполнение. Затем просмотр очереди продолжается со следующего за ним задания в поисках второго. Пример распределения заданий по узлам ВК дан на рис. 1.

На нем представлено параллельное исполнение заданий с ширинами 1, 2, 5 и 8. При этом каждое из заданий может занимать произвольные узлы. Даже при наличии заданий в очереди часть узлов может простаивать. В нашем примере в очереди могут быть задания шириной шесть и более, которым на рассматриваемый момент времени, показанный на рисунке, нет места на исполнение, т.е. нет достаточного числа узлов ВК. Более узкое задание, которому такого места хватает, должно быть извлечено из очереди, несмотря на то что оно пришло позднее стоящих впереди него. Поскольку порядок обслуживания заданий может не соответствовать порядку их прихода (поступления в очередь), то номера заданий на рис. 1 идут не подряд.

Рис. 1. Пример распределения заданий по узлам ВК

Таким образом, среди всех заданий, которые можно поставить на исполнение, приоритет принадлежит пришедшим ранее.

Введем показатели очереди на основе показателей содержащихся в ней заданий. Длиной очереди назовем число находящихся в ней заданий, шириной (сложностью, площадью) очереди — сумму ширин (сложностей, площадей) входящих в нее заданий. Аналогично определим понятия длины, ширины и площади всей вычислительной системы, т. е. ВК.

Также мы столкнулись с одной особенностью. Существуют два средних времени ожидания в очереди. Ввиду того что в очередь попадают не все задания, а лишь их определенный процент, среднее время ожидания можно рассчитывать двумя способами:

1) для всех заданий, учитывая нулевое время ожидания, не попавших в очередь (будем обозначать ее без штриха);

2) только для попавших в очередь (со штрихом).

2. Постановка задачи

В данной работе ставится цель — продемонстрировать построенную модель ВК и основы ее программной реализации на примере предсказания работы кластера UniLu-GAIA при изменении, например, ширины кластера или производительности его узлов. Этот кластер выбран потому, что логи (журналы) его работы есть в открытом доступе [8] и работа может быть повторена (хотя природа исполняемых заданий конфиденциальна и неизвестна). Логи — списки выполненных кластером заданий, где присутствуют как минимум время прихода, длина и ширина заданий. Будет использован лог UniLu-Gaia-2014-2.swf, который принадлежит кластеру шириной 2004 на момент фиксации лога [9].

3. Описание стохастических моделей

В целях стохастического моделирования было усовершенствовано средство, описанное в работах [10, 11]. В качестве допустимой погрешности имитационного моделирования было взято значение 5 % как рекомендуемое во многих исследованиях [12]. В соответствии с Центральной предельной теоремой [12] это требует более 40 испытаний на проверку каждого из вариантов.

Сама процедура моделирования уже реализована нами в названной выше детерминированной модели [10, 11]. Стохастическая модель может использовать уже созданную модель, если заменить детерминированные данные нагрузки стохастическими. Это позволит нам рассмотреть различные варианты прихода заданий, а также заменить большой список входящих заданий простой математической моделью их генерации — моделью входящей нагрузки (МВН).

Каждая модель нагрузки состоит из двух частей: модели времен приходов заданий и модели обслуживания (определение длины, ширины и площади). Обсуждение этих моделей кратко дается в работах [13, 14].

Начнем с самого простого варианта, когда возможны аппроксимация времени прихода заданий *A* и аппроксимации времени обслуживания заданий *B*, *B*, где *A* — аппроксимация некоторым законом распределения интервала между приходами заданий, *B* — аппроксимация законом распределения площади (при указании "^" — длины) заданий. Ширина моделируется отдельно по вероятности ее появления. Зная ширину и длину (площадь), мы можем определить и площадь (длину).

Вместо *A* и *B* будут использованы аппроксимации конкретными законами распределения.

Задания различной ширины могут иметь весьма различные характеристики распределения длины/площади. Поэтому имеет смысл выделить отдельные законы для интервалов ширины [4, 7]. В самом простом случае выделяем для каждой ширины свое распределение длины/площади задания. Таким образом, получаем отдельный поток заданий для каждой из возможных ширин. Обозначим этот вариант значком "$" перед обозначением закона распределения площади: $*B*..

В силу того что в каждом из таких входных потоков будет только одна ширина, длина будет пропорциональна площади и в отдельной аппроксимации не нуждается.

Другой вариант разделения — выделить в отдельную группу каждую ширину, равную степени двойки. Это имеет смысл, так как согласно [5] в логах доминируют именно задания, ширина которых является степенью двойки, — даже тогда, когда к этому нет технических предпосылок. Такие работы, как [6], выделяют еще и другие доминирующие (но слабее, чем степени двойки) ширины заданий, например кратные десяти. В других работах [15], наоборот, пытаются уйти от этой тенденции. Интервалы ширин между степенями двойки выделим также в отдельные группы: по одной группе на каждый интервал. То есть будут группы 1, 2, 3, 4, 5—7, 8, 9—15, 16, 17—31, 32, 33—63, 64 и т.д. Обозначим такое выделение групп знаком "&": *&B* и *&B*.

Практика (как наша, так и [4, 7]) показывает, что при разделении входного потока на несколько потоков по ширине часто можно повысить качество аппроксимации нагрузки; получаемая МВН лучше описывает нагрузку.

Аналогичное разделение можно провести и для входных времен заданий. Выделим несколько входных потоков, в каждый из которых приходят задания с ширинами, относящимися к определенному интервалу. Используем аналогичные принципы разбиения и обозначения. Получатся *&A*, *&A*.

В работе [5] предлагается анализировать входной поток как нестационарный. Необходимость этого была подтверждена в нашей работе [16]. В данной работе будет рассмотрено изменение интенсивности потока заявок в течение недели. За начало недели мы взяли полночь с воскресенья на понедельник. Будем полагать, что интенсивность прихода заданий остается постоянной в течение получаса (как в работе [5]). Это предположение вводится исключительно для упрощения вычисления интегралов, при получении самих формул его не задействуют.

Под приведенной интенсивностью подразумевается

$$\overline{\lambda(t)} = \lambda(t)\overline{\lambda}, \qquad (1)$$

где $\overline{\lambda(t)}$ — приведенная интенсивность прихода заданий; $\lambda(t)$ — интенсивность; $\overline{\lambda}$ — средняя интенсивность в течение недели.

Для генерации интервала между событиями нестационарного потока исказим временную

шкалу. Интервалом времени между реальными точками t_0 и t_1 временной шкалы будем считать значение интеграла

$$L(t,t_0)=\int_{t_0}^{t}\overline{\lambda(t)}dt. \qquad (2)$$

Назовем его "случайным приведенным временем" между приходами заданий. Такой поток будет стационарным, и его можно аппроксимировать обычным способом, а потом вернуться к изначальной шкале времени.

Обозначим такую модель знаком "~" перед обозначением входного потока, например, *~A*. Здесь сразу же надо оговорить одну важную особенность. Обозначения ~$*A* и $~*A* не одинаковы. В первом случае подразумеваем введение единой интенсивности для всех входных потоков, а во втором — что каждый поток получает свою собственную интенсивность.

Таким образом, получаем следующие варианты приходов: *A*, *~A*, $*A*, &*A*, ~$*A*, ~&*A*, $~*A*, &~*A* и варианты обслуживания: *B*, $*B*, &*B*, *B*, &*B*. Сочетание этих двух моделей дает модель нагрузки. Будем обозначать его (сочетание) через знак слеша, например &~*A*/&*B*. Итого здесь имеем 40 комбинаций приходов заданий и параметров их обслуживания.

4. Законы распределений

Введем обозначения: $E(X)$ — математическое ожидание величины X, $VAR(X)$ — ее дисперсия, $stDev(X)$ — среднее квадратичное отклонение, $cov(X)=\dfrac{stDev(X)}{E(X)}$ — коэффициент вариации. Если мы имеем дело с оценкой момента, то будем писать над ней горизонтальную черту.

Методами аппроксимации являются метод моментов (ММ) и метод наибольшего правдоподобия (МНП). Вариант ММ предполагает определение параметров распределения по моментам, т. е. число оценок моментов должно равняться числу параметров распределения. Вариант МНП подразумевает максимизацию функции правдоподобия, которая часто выполняется численно.

В качестве законов распределения случайных после уточнения выводов [13, 14] величин были взяты следующие (в различных источниках встречаются разные обозначения, поэтому приводим используемые нами обозначения).

1. M — экспоненциальное распределение:

$$pdf(x)=\lambda\mathbf{e}^{-\lambda x};\ cdf(x)=1-\mathbf{e}^{-\lambda x};$$
$$E(X)=stDev(X)=\frac{1}{\lambda}.$$

Для этого распределения оценки ММ и МНП совпадают и дают

$$\lambda=\frac{1}{\overline{E(X)}}.$$

Решение для МНП возможно аналитически.

2. Г — гамма-распределение:

$$pdf(x)=\lambda\frac{(\lambda x)^{v-1}}{\Gamma(v)}\mathbf{e}^{-\lambda x};$$

$$cdf(x)=\frac{\gamma(v,\lambda x)}{\Gamma(v)}=P(v,\lambda x);$$

$$E(X)=\frac{v}{\lambda};\ VAR(X)=\frac{v}{\lambda^2},$$

где $\Gamma(x)$ — гамма-функция Эйлера; $\gamma(x, y)$ — неполная нижняя гамма-функция; $P(x, y)$ — нормированная нижняя гамма-функция.

Оценки ММ (Γ_μ) имеют вид

$$\lambda=\frac{\overline{E(X)}}{\overline{VAR(X)}};\ \ v=\lambda\overline{E(X)}.$$

Оценки МНП (Γ_λ) найти труднее. Из равенства частных производных логарифмической функции правдоподобия нулю удалось получить

$$\lambda=\frac{v}{\overline{E(X)}},\ \ \Psi(v)=\overline{E(\ln X)}-\ln(\overline{E(X)})+\ln v,$$

где $\Psi(v)$ — дигамма-функция; $\overline{E(\ln X)}$ — оценка среднего значения логарифма случайной величины.

Стоит отметить, что обе оценки дают одно и то же математическое ожидание, которое совпадает с оценкой ($E(X)=\overline{E(X)}$), но остальные моменты будут различаться в общем случае.

Все указанные выше распределения будем считать простыми в противоположность гиперраспределениям, которые следуют далее.

3. $H(n)$ — гиперэкспоненциальное распределение:

$$pdf(x)=\sum_{i=1}^{n}\alpha_i\lambda_i\mathbf{e}^{-\lambda_i x};\ \ cdf(x)=1-\sum_{i=1}^{n}\alpha_i\mathbf{e}^{-\lambda_i x};$$

$$1\geqslant\alpha_i\geqslant 0;\ \sum_{i=1}^{n}\alpha_i=1;\ cov(X)\geqslant 1,$$

где n — число ветвей, веток распределения (задается перед аппроксимацией как часть вида распределения).

В нашей работе [17] было введено упрощение этого распределения для ММ в двуветочном случае ($n = 2$). Обозначим его H_{μ}. МНП с численным решением применим в любом случае, поэтому обозначим этот вариант как $H\Gamma(2)$, где n — число ветвей, веток распределения.

4. $H\Gamma(n)$ — гипергамма-распределение:

$$pdf(x) = \sum_{i=1}^{n} \alpha_i \lambda_i \frac{(\lambda_i x)^{v_i - 1}}{\Gamma(v_i)} \mathbf{e}^{-\lambda_i x};$$

$$cdf(x) = \sum_{i=1}^{n} \alpha_i P(v_i, \lambda_i x); \quad \sum_{i=1}^{n} \alpha_i = 1,$$

$$1 \geqslant \alpha_i \geqslant 0, \; cov(X) \in (0; \infty),$$

где n — число ветвей, веток распределения (задается перед аппроксимацией как часть вида распределения).

В нашей работе [18] были введены два упрощения для двуветочного случая: $H\Gamma_{\mu}$ для ММ и $H\Gamma_{\lambda}$ для МНП. МНП с численным решением применим в любом случае, поэтому обозначим этот вариант как $H\Gamma(n)$, где n — число ветвей, веток распределения.

5. Расширяемость программной разработки

Представленная модель реализована в виде программной разработки [19], которая построена с использованием паттернов программирования, что позволяет быстро расширять написанную модель без серьезных изменений программного кода. В качестве языка реализации выбран язык Java как сочетание производительности и удобности разработки. Основными паттернами для нас являются Адаптер, Команда, Стратегия, Фабрика [20].

На рис. 2 дана UML-диаграмма характеристик кластера. В нашем случае для характеристик используется класс NodeChar. Он содержит в себе два интерфейса.

Интерфейс TaskChooser отвечает за выбор задания из очереди, постановку задачи на очереди или исполнение. Таким образом, описанная нами процедура работы с очередью является лишь одной из реализаций этого интерфейса. Создав другую реализацию, можно реализовать свою собственную политику обработки очереди.

Второй интерфейс TaskStreamFactory отвечает за создание конкретного входящего потока заданий. Фабрика необходима, так как при стохастическом моделировании рассматривается большое число исходов и необходимо произвольное число потоков заданий. Сам поток представлен интерфейсом TaskStream. Использование интерфейсов позволяет при необходимости создать любой свой собственный поток входящих заданий, не ограничиваясь не только нашими МВН, но и подходом к генерации нагрузки в целом (например, она может поступать из готового файла без вмешательства программы).

На рис. 3 представлена UML-диаграмма TaskStreamFactory, а на рис. 4 — UML-диаграмма производимого фабрикой интерфейса TaskStream. В нашей реализации этой пары DefaultTaskStreamFactory производит Default-

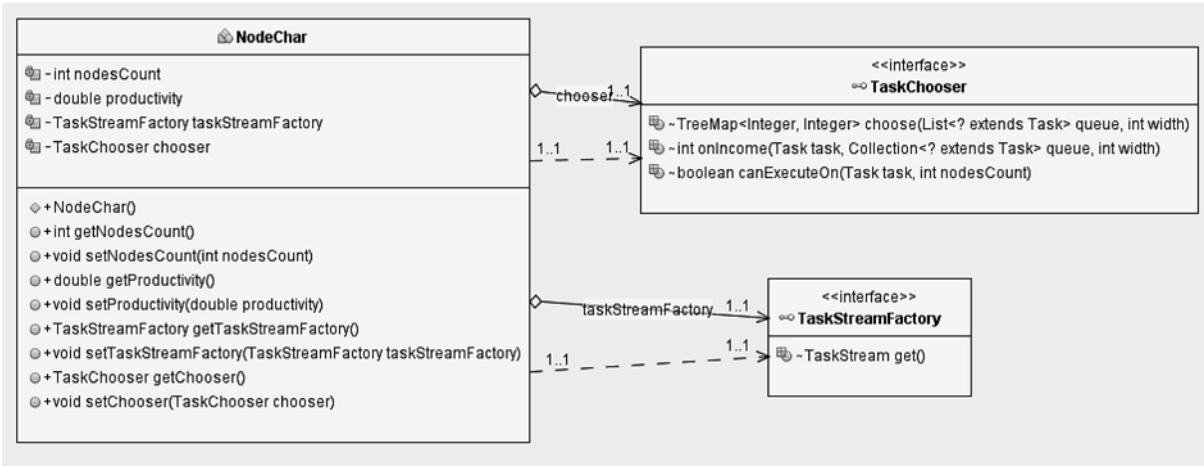

Рис. 2. UML-диаграмма характеристик узла (NodeChar)

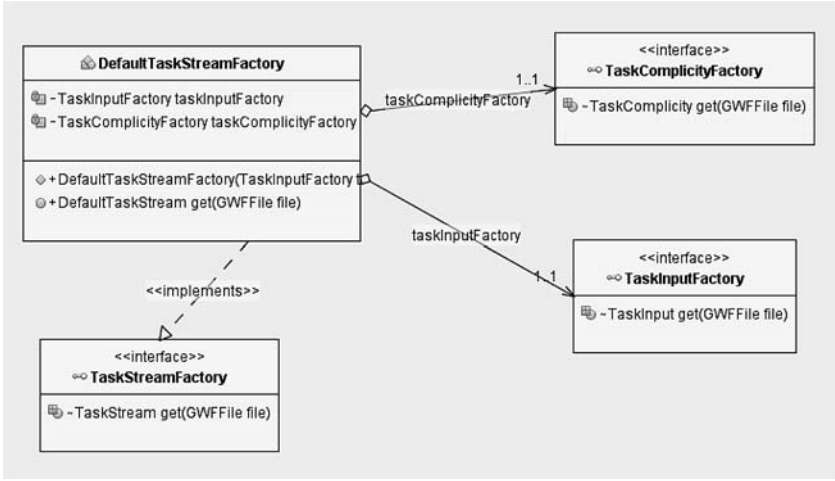

Рис. 3. UML-диаграмма TaskStreamFactory

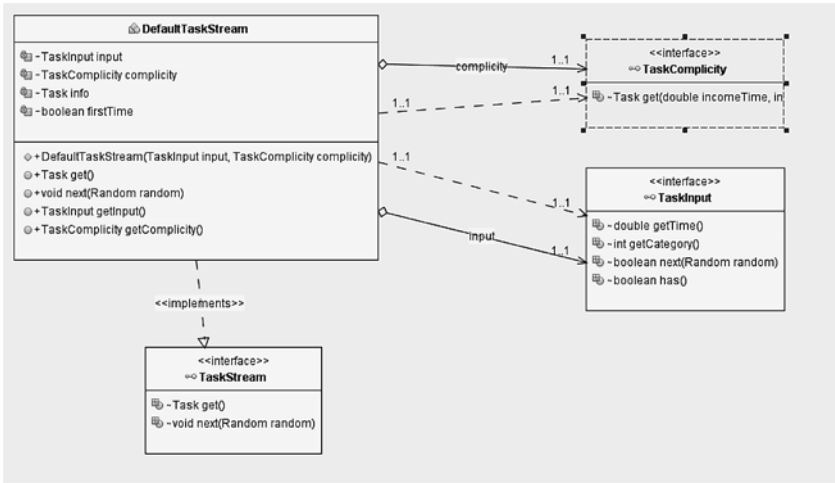

Рис. 4. UML-диаграмма TaskStream

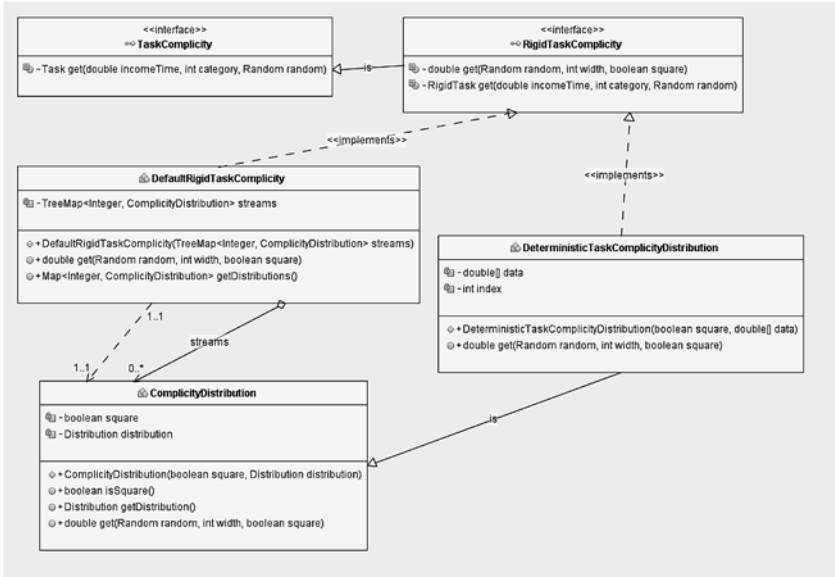

Рис. 5. UML-диаграмма интерфейса TaskComplicity

TaskStream: вся МВН делится на модели прихода заданий и модель обслуживания, и именно эти классы выполняют объединение.

Модель прихода соответствует паре интерфейсов TaskInputFactory и TaskInput, а модель обслуживания — паре TaskComplicityFactory и TaskComplicity. Написав свою реализацию для любой пары интерфейсов, можно воспроизвести свои модели прихода или обслуживания.

На рис. 5 показаны реализации TaskComplicity. В настоящий момент все реализации сводятся к реализации RigidTaskComplicity, который оперирует немасштабируемыми заданиями. При необходимости код программы может быть быстро расширен, например, интерфейсом MoldableTaskComplicity с реализациями для масштабируемых заданий по тем правилам масштабирования, которые нужны реализующему.

6. Моделирование

После проверки модели были найдены несколько наиболее подходящих МВН для кластера UniLu-GAIA [21]. В данной работе мы выберем модель &~H(2)/Γλ^.

Сначала рассмотрим альтернативные вариации ширины кластера UniLu-GAIA. Максимальная ширина задания, которая встречается в логах данного кластера, — 516. Значит, минимальная ширина кластера для обслуживания этого лога — тоже 516. Поэтому вполне резонно должен быть задан вопрос, не является ли текущая ширина кластера 2004 избыточной или, наоборот, необходимо нарастить ширину кластера, потому что попавшие в очередь задания ждут там очень долго.

Используем моделирование, чтобы определить это. Дадим небольшой запас кластеру по ширине.

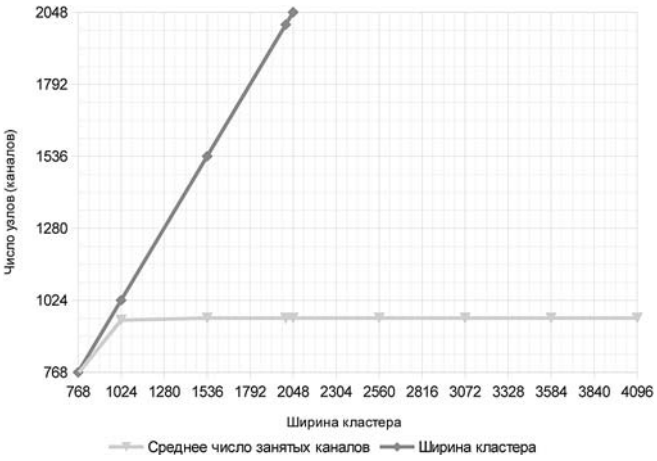

Рис. 6. Использование каналов кластера UniLu-GAIA при различной ширине кластера

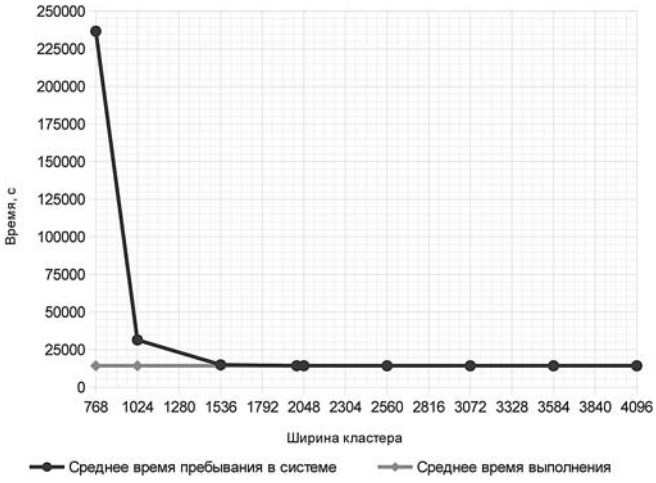

Рис. 7. Среднее время пребывания заданий в кластере UniLu-GAIA при различной ширине кластера

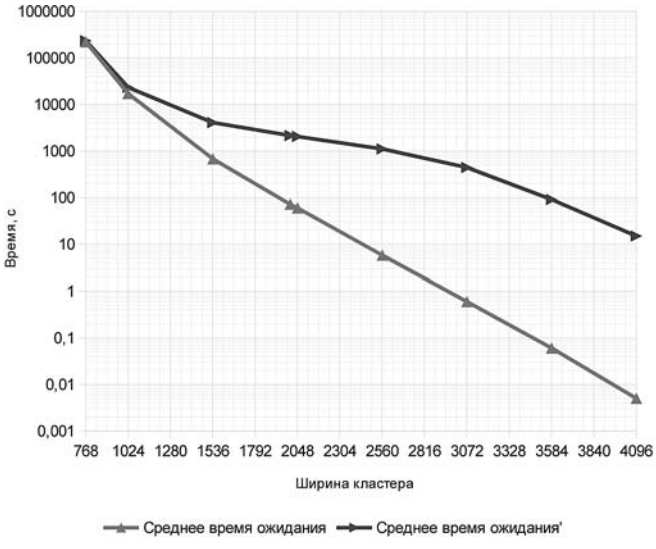

Рис. 8. Соотношение времен ожидания

Будем ориентировать на степени двойки, поэтому минимальной шириной будет 768. Также рассмотрим 1024, 1536, 2048, 2560, 3072, 3584, 4096.

На рис. 6 показано использование кластера при различной ширине.

Очевидно, что кластеру необходимы около 960 каналов, чтобы обслужить входящий поток заданий. В противном случае он просто не будет успевать обслуживать все задания. Это требование накладывается самим потоком входящих заданий.

Наиболее важным параметром для нас является среднее время пребывания в системе. Именно этот параметр говорит, сколько времени прошло с момента отправки задания на исполнение до получения результата. Соотношение времени ожидания в очереди и времени выполнения интересует нас лишь опосредованно.

Согласно рис. 7 при ширине кластера примерно до 960 наблюдается резкое увеличение времени пребывания в системе, так как кластер просто не справляется с таким потоком заданий.

Где-то при ширине примерно 1500 наблюдается практически совпадение графиков времени пребывания в системе и времени выполнения. Значит, среднее время ожидания всех заданий становится пренебрежимо малым по сравнению со временем обслуживания. Время выполнения составляет около 4...4,5 ч (изменить время выполнения изменением ширины кластера нельзя, так как задания не масштабируются).

Но даже при таком малом среднем значении времени ожидания всех заданий среднее время ожидания, взятое лишь для заданий, попавших в очередь, может быть сопоставимым со временем обслуживания. Иными словами, если задание попадает в очередь, оно попадает туда надолго. На рис. 8 представлены эти соотношения.

При ширине примерно в 1500 среднее время ожидания всех заданий составляет около 11 мин. При времени выполнения 4...4,5 ч им действительно можно пренебречь, но при этом среднее время ожидания заданий, попавших в очередь, составляет около 65 мин, что уже около четверти времени исполнения и отброшено быть не может. В такой ситуации в очередь попадает примерно каждое шестое задание (рис. 9). Значит, каждое шестое задание ждет около часа, а остальные пять исполняются сразу. Отсюда и среднее время ожидания всех заданий 11 мин. Поэтому и необходимы

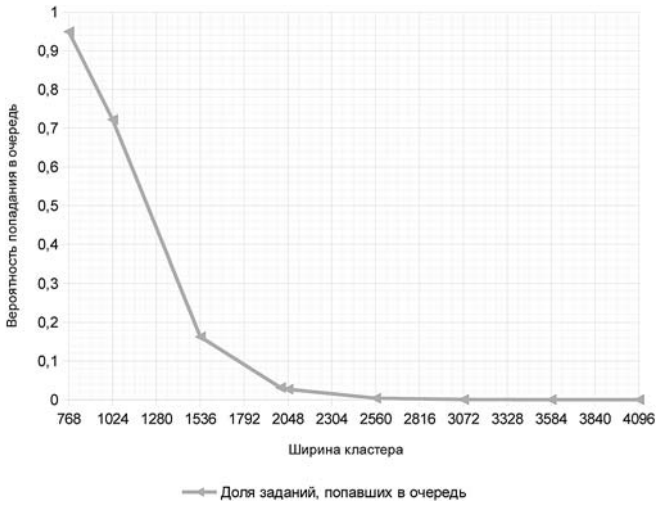

Рис. 9. Вероятность попадания в очередь

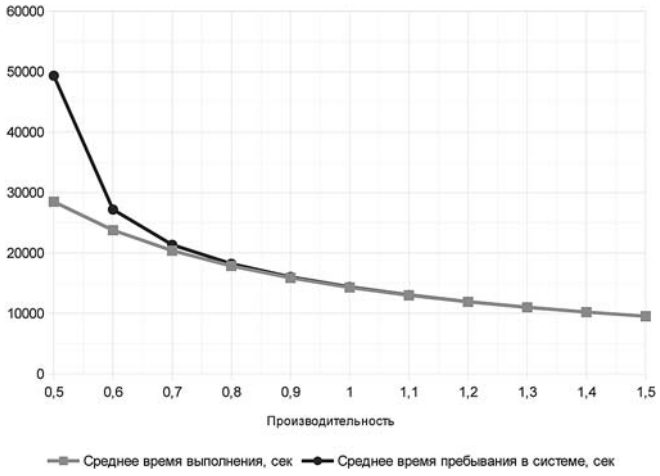

Рис. 10. Зависимость временных характеристик задания от производительности

два времени ожидания, чтобы оценить происходящее.

Увеличение ширины кластера до 2004 сокращает среднее время ожиданий попавших в очередь до 36 мин. При этом в очередь попадает примерно каждое 30-е задание, так что в среднем задания бывают в очереди около 1...1,3 мин. И администраторы кластера выбрали именно эту ширину.

Для сравнения наращивание ширины примерно до 3000 сократит среднее время ожиданий попавших в очередь до 7,5 мин, при этом в очередь попадет каждое 2000-е задание и среднее время ожидания всех заданий составит меньше секунды, но это на 50 % дороже по узлам. При ширине примерно в 4000 задания, попавшие в очередь, ждут там в среднем 15 с. Два других показателя слишком малы, чтобы их можно было учитывать.

Таким образом, наилучшей шириной кластера следует выбрать число, близкое к 2000. На ширинах кластера до 1000 кластер даже не способен справиться с потоком заданий, поэтому их нельзя рассматривать.

Следующий важный вопрос — это определение производительности узлов кластера. За единицу производительности возьмем текущую производительность кластера и отметим увеличение/уменьшение этой величины (на графике — доли единицы). Здесь делается допущение, что у всех заданий времена выполнения будут изменяться обратно пропорционально производительности. В общем случае такое допущение неверно, но без знания природы заданий невозможно отказаться от него.

Ответ на вопрос подходящей производительности также способно дать имитационное моделирование. В качестве примера будем снова рассматривать кластер UniLu-GAIA. На рис. 10 изображено изменение параметров обслуживания в зависимости от производительности. За единичную производительность взята текущая производительность кластера.

В данном случае уже нет такого явного порога изменения характеристик, который был при изменении ширины кластера. Но можно понять, что где-то при производительности, равной 0,7—0,8 от текущей и выше, время ожидания в очереди становится пренебрежимо малым по сравнению со временем обслуживания. Рекомендовать какую-то конкретную производительность здесь нельзя: все определяется финансовыми и техническими возможностями организации, содержащей кластер.

Заключение

Таким образом, было создано средство моделирования вычислительных кластерных систем, которое позволяет получить характеристики обслуживания и предсказать их для некоторой конфигурации [22]. Средство является расширяемым за счет использования паттернов программирования, в частности выделения необходимых интерфейсов.

В качестве примера было проведено моделирование работы кластера UniLu-GAIA, на котором были показаны конкретные характеристики обслуживания и приняты конкретные решения.

Список литературы

1. **Sinisterra M. M., Henao T. M. D., López E. G. R.** Clúster de balanceo de carga y alta disponibilidad para servicios web y mail. URL: https://dialnet.unirioja.es/descarga/articulo/4364562.pdf (дата обращения 04.01.2018).
2. **Аветисян А. И., Гайсарян С. С., Грушин Д. А., Кузюрин Н. Н., Шокуров А. В.** Эвристики распределения задач для брокера ресурсов Grid. URL: http://www.citforum.ru/nets/digest/grid/index.shtml (дата обращения 04.01.2018).
3. **GridMe:** Grid modeling environment. Google code. URL: https://code.google.com/p/gridme/ (дата обращения 04.01.2018).
4. **Jann J., Pattnaik P., Franke H., Wang F., Skovira J., Riordan J.** Modeling of Workload in MPPs // Job Scheduling Strategies for Parallel Processing: Lect. Notes Comput. Sci. Springer-Verlag. 1997. Vol. 1291. P. 95—116.
5. **Lublin U., Feitelson D. G.** The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs // The Rachel and Selim Benin School of Computer Science and Engineering. URL: http://www.cs.huji.ac.il/~feit/papers/Rigid01TR.pdf (дата обращения 04.01.2018).
6. **The Feitelson** 1996 Model // The Rachel and Selim Benin School of Computer Science and Engineering. URL: http://www.cs.huji.ac.il/labs/parallel/workload/m_feitelson96/ (дата обращения 04.01.2018).
7. **The Jann** et al 1997 Model // The Rachel and Selim Benin School of Computer Science and Engineering. URL: http://www.cs.huji.ac.il/labs/parallel/workload/m_jann97/ (дата обращения 04.01.2018).
8. **Logs** of Real Parallel Workloads from Production Systems // The Rachel and Selim Benin School of Computer Science and Engineering. URL: http://www.cs.huji.ac.il/labs/parallel/workload/logs.html (дата обращения 04.01.2018).
9. **HPC** @ Uni.lu. URL: https://hpc.uni.lu/systems/gaia/ (дата обращения 04.01.2018).
10. **Гаевой С. В., Аль-Хадша Ф. А. Х., Лукьянов В. С.** Детерминированная имитационная модель кластеров грид-системы, обслуживающих задания // Вестник компьютерных и информационных технологий. 2014. № 6. С. 39—43.
11. **Гаевой С. В., Аль-Хадша Ф. А. Х., Фоменков С. А., Лукьянов В. С.** Детерминированная имитационная модель кластеров грид-системы для сравнения эффективности использования эвристик распределения заданий // Прикаспийский журнал: управление и высокие технологии. 2014. № 2. С. 148—157.
12. **Фоменков С. А., Камаев В. А., Орлова Ю. А.** Математическое моделирование системных объектов: учеб. пособ. (гриф). Доп. УМО вузов по университетскому политехн. образованию. Волгоград, 2014. 335 с.
13. **Гаевой С. В., Аль-Хадша Ф. А. Х., Фоменков С. А.** Аппроксимация времени выполнения заданий на примере вычислительного кластера LPC EGEE 2004 // Известия ВолгГТУ. Серия "Актуальные проблемы управления, вычислительной техники и информатики в технических системах". 2014. № 12 (139). С. 135—141.
14. **Гаевой С. В., Аль-Хадша Ф. А. Х.** Моделирование работы вычислительного кластера на примере LANL CM5 // SCI-ARTICLE.RU: электронный периодический научный журнал. 2013. № 3 (ноябрь). С. 304—313. URL: http://sci-article.ru/stat.php?i=modelirovanie_rabotY_vYchislitelnogo_klastera_na_primere_LANL_CM5 (дата обращения 04.01.2018).
15. **Downey A. B.** A Parallel Workload Model and Its Implications for Processor Allocation // The Rachel and Selim Benin School of Computer Science and Engineering. URL: http://allendowney.com/research/allocation/ (дата обращения 04.01.2018).
16. **Ахмед В. М. А., Гаевой С. В., Фоменков С. А.** Влияние нестационарности входного потока заданий на обслуживание в кластерной системе // Вестник компьютерных и информационных технологий. 2017. № 11 (161). С. 44—52.
17. **Гаевой С. В., Ахмед В. М. А., Быков Д. В., Фоменков С. А.** Сокращение времени аппроксимации логов вычислительного кластера с использованием методов моментов на гиперэкспоненциальном распределении // Прикаспийский журнал: управление и высокие технологии. 2017. № 1. С. 94—105.
18. **Ахмед В. М. А., Гаевой С. В., Фоменков С. А.** Сокращение времени аппроксимации нагрузки вычислительного кластера с использованием упрощения гипер-гамма-распределения // Вестник Воронежского института высоких технологий. 2017. № 4 (23). С. 52—58.
19. **Гаевой С. В., Ахмед В. М. А., Фоменков С. А.** ВолгГТУ. Средство аппроксимации и имитационного моделирования вычислительных нагрузок (SWFJParser. JDSBrocker). Свидетельство о гос. регистрации программы для ЭВМ № 2017619355 от 24 августа 2017 г. Российская Федерация. 2017.
20. **Freeman E., Bates B., Sierra K., Robson E.** Head First Design Patterns. A Brain-Friendly Guide. O'Reilly Media. 2009. 688 p.
21. **Гаевой С. В., Ахмед В. М. А., Быков Д. В., Фоменков С. А.** Аппроксимация потока заданий на примере вычислительного кластера UniLu-Gaia // Известия ВолгГТУ. Серия "Актуальные проблемы управления, вычислительной техники и информатики в технических системах". 2017. № 8 (203). С. 96—102.
22. **Gaevoy S. V., Ahmed W. M. A., Fomenkov S. A., Kolesnikov S. G.** Methods of Modeling Incoming Jobs Stream used on the Computing Cluster UniLu-Gaia // Journal of Engineering and Applied Sciences. 2017. Vol. 12, Issue 24. P. 7548—7554. URL: http://medwelljournals.com/abstract/?doi=jeasci.2017.7548.7554. DOI: 10.3923/jeasci.2017.7548.7554

S. V. Gaevoy, Ph. D. (Engineering), Senior Lecturer, e-mail: gaevserge@mail.ru;
W. M. A. Ahmed, Post-Graduate Student, e-mail: wesamalsofi@gmail.com;
S. A. Fomenkov, D. Sc. (Engineering), Professor, Professor of "CAD" department, e-mail: saf@vstu.ru,
Volgograd State Technical University, Volgograd, 400005, Russian Federation

Using Simulation to Determine Quality of Service of a Cluster System

In this paper computing clusters (CC) are considered. They are used to execute incoming jobs. There is such a CC in our university and we need to predict its service characteristics at executing several workloads. An important method to analyze parallel workloads is modeling execution of those systems by using parallel workload models (PWM). A simulation model of cluster has already been created at our department. We have already proposed many PWMs, but all these PWMs use

a continuous variable approximation. This approximation can be done either by method of moments (MM), or maximum likelihood method (MLM). The latter gives the more accurate results but consumes much time. It was empirically proved in our and third-party papers. In this paper the goal is to demonstrate the already built CC model and the basics of its implementation. The implementation uses software design patterns and can adapted to many computing systems. We have chosen Java language because it has good performance and is convenient for the development. The program is cross-platform. Everyone can use, for example, Open JDK to compile it for free. The parallel workload of a cluster UniLu-GAIA is used because it is in free access and thus the computing can be repeated by everyone (although the nature of the tasks being executed there is confidential and unknown). The log being used belongs to the cluster with width 2004 at the moment of executing.

Keywords: *parallel workload, parallel workload model, rigid job, job length, simulation, stochastic approximation, quality of service, computing cluster, simulation utility, workload approximation, continuous variable distribution, design pattern*

DOI: 10.17587/it.24.464-474

References

1. **Sinisterra M. M.** [et al.]. Cluster de balanceo de carga y alta disponibilidad para servicios web y mail [Cluster of load balancing and high availability for web and mail services]. Available at: https://dialnet.unirioja.es/descarga/articulo/4364562.pdf

2. **Avetisyan A. I.** [et al.]. *Evristiki raspredeleniya zadach dlya brokera resursov Grid* [Heuristics of job distribution for Grid resource brocker]. Available at: http://www.citforum.ru/nets/digest/grid/index.shtml (in Russian).

3. **GridMe:** Grid modeling. Available at: https://code.google.com/p/gridme/

4. **Jann J.** [et al.]. Modeling of Workload in MPPs. Job Scheduling Strategies for Parallel Processing: Lect. Notes Comput. Sci. ed. by D. G. Feitelson, L. Rudolph. Springer-Verlag, 1997, vol. 1291. P. 95—116.

5. **Lublin U., Feitelson D. G.** The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs. The Rachel and Selim Benin School of Computer Science and Engineering. Available at: http://www.cs.huji.ac.il/~feit/papers/Rigid01TR.pdf

6. **The Feitelson** 1996. The Rachel and Selim Benin School of Computer Science and Engineering. Available at: http://www.cs.huji.ac.il/labs/parallel/workload/m_feitelson96/

7. **The Jann** et al 1997 Model. The Rachel and Selim Benin School of Computer Science and Engineering. Available at: http://www.cs.huji.ac.il/labs/parallel/workload/m_jann97/

8. **Logs** of Real Parallel Workloads from Production Systems. The Rachel and Selim Benin School of Computer Science and Engineering. Available at: http://www.cs.huji.ac.il/labs/parallel/workload/logs.html

9. **HPC** @ Uni.lu. Available at: https://hpc.uni.lu/systems/gaia/

10. **Gaevoy S. V., Al-Hadsha F. A. H., Luk'yanov V. S.** *Determinirovannya imitatsionnaya model klusterov grid-sistemy, obsluzhivayustchikh zadaniya* [Deterministic simulation model of clusters of a Grid-system executing jobs]. *Vestnik komp'iuternykh i informatsionnykh tekhnologiy* [Herald of computer and information technologies], 2014, no. 6, pp. 39—43 (in Russian).

11. **Gaevoy S. V., Al-Hadsha F. A. H., Fomenkov S. A., Luk'yanov V. S.** *Determinirovannaya imitsyonnaya model' klusterov grid-sistemy dl'ya sravnenia effectivnosti ispol'sovaniya evristik raspredeleniya zadaniy* [Deterministic simulation model of clusters of a Grid-system for comparison of heuristics for task distribution]. *Prikaspiyskiy zhurnal: upravlenie i vysokie tekhnologii* [Caspian Journal: Control and High Technologies], 2014, no. 2, pp. 148—157 (in Russian).

12. **Fomenkov S. A., Kamaev V. A., Orlova Yu. A.** Matematicheskoye modelirovanie sistemnykh ob'yektov [Mathematical modeling of system objects], Volgograd, 2014. 335 p. (in Russian).

13. **Gaevoy S. V., Al-Hadsha F. A. H., Fomenkov S. A.** Approksimatsia vremeni vypolneniya zadaniy na primere vychislitilel'nogo klustera LPC EGEE 2004 [Approximation of job execution time discovering computing cluster LPC EGEE 2004]. Izvestiya VolgGTU, seriya "Aktualnye problemy upravleniya, vychislitelnoy tekhniki i informatiki v tekhnicheskikh sistemakh" [Newspaper of VSTU, Series "Actual Problems of Management, Computing Hardware and Informatics in Engineering Systems"], Volgograd, 2014, no. 12 (139), pp. 135—141 (in Russian).

14. **Gaevoy S. V., Al-Hadsha F. A. H.** Modelirovanie raboty vychislitel'nogo klustera na primere LANL CM5 [Simulation of computing cluster discovering LANL CM5]. SCI-ARTICLE.RU: electronnyy periodicheskiy nauchnyy zhurnal [SCI-ARTICLE.RU: Electronic periodical scientific magazine], 2013, no. 3 (Nov), pp. 304—313. Available at: http://sci-article.ru/stat.php?i=modelirovanie_rabotY_vYchislitelnogo_klastera_na_primere_LANL_CM5 (in Russian).

15. **Downey A. B.** A Parallel Workload Model and Its Implications for Processor Allocation. The Rachel and Selim Benin School of Computer Science and Engineering. Available at: http://allendowney.com/research/allocation/

16. **Ahmed W. M. A., Gaevoy S. V., Fomenkov S. A.** *Vlijanie nestacionarnosti vhodnogo potoka zadanij na obsluzhivanie v klasternoj sisteme* [Incoming Job Streams Nonstationary Nature Influence Streams on the Service in a Cluster System]. *Vestnik komp'iuternykh i informatsionnykh tekhnologiy* [Herald of computer and information technologies], 2017, no. 11 (161), pp. 44—52 (in Russian).

17. **Gaevoy S. V., Ahmed W. M. A., Bykov D. V., Fomenkov S. A.** *Sokrashhenie vremeni approksimacii logov vychislitel'nogo klastera s ispol'zovaniem metodov momentov na giperjeksponencial'nom raspredelenii* [Reducing the approximation time of cluster workload by using method of moments on Hyperexponential distribution]. *Prikaspiyskiy zhurnal: upravlenie i vysokie tekhnologii* [Caspian Journal: Control and High Technologies], 2017, no. 1, pp. 94—105 (in Russian).

18. **Ahmed W. M. A., Gaevoy S. V., Fomenkov S. A.** *Sokrashhenie vremeni approksimacii nagruzki vychislitel'nogo klastera s ispol'zovaniem uproshhenija giper-gamma-raspredelenija* [Reducing the Approximation Time of Cluster Workload by Using Simplified Hypergamma Distribution]. *Vestnik Voronezhskogo instituta vysokih tehnologij* [Herald of Voronezh Institute of High Technologies], 2017, no. 4 (23), pp. 52—58 (in Russian).

19. **Gaevoy S. V., Ahmed W. M. A., Fomenkov S. A.** VSTU. *Svid. o gos. registracii programmy dlja JeVM № 2017619355 ot 24 avgusta 2017 g. Rossijskaja Federacija. Sredstvo approksimacii i imitacionnogo modelirovanija vychislitel'nyh nagruzok (SWFJParser. JDSBrocker)* [Certificate of state registration of the computer program no. 2017619355. August 24, 2017. Russian Federation. Utility for approximation and simulation of parallel workloads (SWFJParser.JDSBrocker)] (in Russian).

20. **B. Bates** etc. Head First Design Patterns. A Brain-Friendly Guide. O'Reilly Media, 2009, 688 p.

21. **Gaevoy S. V., Ahmed W. M. A., Bykov D. V., Fomenkov S. A.** *Approksimacija potoka zadanij na primere vychislitel'nogo klastera UniLu-Gaia* [The Approximation of Task Stream by Using the Computing Cluster Unilu-Gaia]. *Izvestiya VolgGTU, seriya "Aktualnye problemy upravleniya, vychislitelnoy tekhniki i informatiki v tekhnicheskikh sistemakh"* [Newspaper of VSTU, Series "Actual Problems of Management, Computing Hardware and Informatics in Engineering Systems"], Volgograd, 2017, no. 8 (203), pp. 96—102 (in Russian).

УДК 004.94 DOI: 10.17587/it.24.474-480

О. С. Исаева, канд. техн. наук, ст. науч. сотр., e-mail: isaeva@icm.krasn.ru,
Институт вычислительного моделирования Сибирского отделения Российской академии наук — обособленное подразделение ФИЦ КНЦ СО РАН, Красноярск

Технология построения комплексных моделей в инфраструктуре имитационного моделирования

Предложена технология построения комплексных имитационных моделей, позволяющая объединить модели, построенные по стандарту SMP2 (Simulation model portability), виртуальные приборы и логические модели, методы функционирования которых задаются в базах знаний. Технология основана на формализации моделей, содержащей структурно-параметрическое и функциональное представления. Она позволяет формировать семантические конструкции в терминах предметной области, что обеспечивает удобство построения моделей и наглядность представления моделируемой системы в целом. Технология апробирована в программном обеспечении, предназначенном для имитационного моделирования спутниковых систем.

Ключевые слова: *космический аппарат, бортовая аппаратура, имитационное моделирование, стандарт Simulation model portability (SMP2), инфраструктура имитационного моделирования*

Введение

Разработка сложных проектов в аэрокосмической отрасли предполагает обеспечение высокого уровня взаимодействия различных производителей. В случае европейских проектов кооперация основывается на стандартах Европейского космического агентства, которые задают требования к нормативным документам, информационным базам, программным компонентам для всех этапов жизненного цикла производства космической техники [1]. Стандарты определяют существенную роль моделирования при реализации космических проектов. Принципы имитационного моделирования технических объектов, состоящих из подсистем, изготавливаемых разными производителями, заложены в стандарте "Simulation Model Portability" (текущая редакция SMP2) [2]. Системы моделирования, реализующие стандарт SMP2, называют инфраструктурами имитационного моделирования (Simulation Infrastructure). Существует целый ряд инфраструктур моделирования, в их числе: SimSAT — Европейского космического агентства [3], SimTG — Astrium Satellites [4], симулятор центра управления полетами SWARMSIM [5] и др.

Интеграция моделей и технологий, построенных на основе SMP2, выполняется в различных научно-исследовательских проектах. Примером успешного применения SMP2 является франко-германский проект разработки спутников дистанционного зондирования для мониторинга парникового газа — MERLIN [6]. Проект объединяет несколько компаний, в их числе: Airbus, CNES и Thales Alenia Space. Для реализации проекта выполнена интеграция имитационных моделей, разрабатываемых исследователями на четырех различных SMP2 платформах имитационного моделирования. Применение в данном проекте SMP2 позволило выполнить стандартизацию всех компонентов пространственной системы, сделать их более надежными, снизить затраты и оптимизировать график разработки. Положительный опыт подобных интеграций показывает актуальность и обоснованность ориентации программного обеспечения имитационного моделирования на стандарт SMP2.

В отличие от существующих решений, основанных на стандарте SMP2, для имитаци-