УДК 004.021

СТРУКТУРНО-ФУНКЦИОНАЛЬНЫЙ АНАЛИЗ СЕРВИС-ОРИЕНТИРОВАННЫХ СИСТЕМ

А.Ю. Шелестов[1], О.М. Куссуль[2], Б.Я. Яйлимов[1]

[1] *Институт космических исследований НАН Украины и ГКА Украины,*
[2] *Национальный технический университет Украины «Киевский политехнический институт»*

inform@ikd.kiev.ua, olgakussul@gmail.com, yailymov@gmail.com

Проаналізовано особливості сервіс-орієнтованої системи на основі структурно-функціонального аналізу. Представлені результати чисельного моделювання продуктивності сервіс-орієнтованої системи.
Ключові слова: сервіс-орієнтована система, структурно-функціональний аналіз, сервіс-орієнтована архітектура

This paper analyzes basic features of service-oriented systems using structural-functional analysis. The results of numerical calculations of service-oriented systems performance determination are presented.
Keywords: service-oriented system, structural-functional analysis, service-oriented architecture

Проанализированы особенности сервис-ориентированной системы на основе структурно-функционального анализа. Представлены результаты численного моделирования производительности сервис-ориентированной системы.
Ключевые слова: сервис-ориентированная система, структурно-функциональный сервис-ориентированная архитектура

Введение

Стремительное развитие компьютерных технологий привело к тому, что предприятиям становится очень дорого строить собственные подразделения со своей инфраструктурой для выполнения текущих задач, поэтому образуются объединения предприятий, которые используют одну общую инфраструктуру. Поэтому при построении распределенных систем преобладает сервис-ориентированный подход, который оказывается эффективным для решения научных задач на основе высокопроизводительных вычислений, поддержки принятия решений на различных уровнях управления и т.д.

Для реализации сервис-ориентированных систем требуется участие большого количества поставщиков ресурсов, каждый из которых обеспечивает одну и ту же услугу для большого количества запросов.

Сервис-ориентированная архитектура представляет собой парадигму, которая предназначена для проектирования, разработки и управления дискретными единицами логики (сервисами) в вычислительной среде [1]. Применение такого подхода требует от разработчиков проектирования своих приложений как набора сервисов. Разработчики при построении новых компонентов системы могут использовать уже существующие блоки, которые

будут легко интегрироваться в любую систему независимо от деталей реализации.

Сервис - это абстрактное представление конкретных приложений или баз данных, определенных с точки зрения их функционального назначения (выполняемых ими функций). Назначением сервис-ориентированной архитектуры является предоставление пользователям возможности получать лучшие в своем классе компоненты, и не привязываться к единственному поставщику ресурсов. В качестве сервиса может выступать и инфраструктура, и платформа, и программное обеспечение в зависимости от задач, которые ставит перед собой пользователь [2].

Инфраструктура как сервис (Infrastructure as a Service - IaaS) - это предоставление компьютерной инфраструктуры (как правило, в форме виртуализации) как сервиса, доступ к которой реализован на основе концепции облачных вычислений. IaaS охватывает сервисы, которые можно предлагать пользователям на основе принципа почасовой оплаты. Это могут быть такие сервисы, как предоставление вычислительных и сетевых ресурсов, а также ресурсов хранения данных.

Платформа как сервис (Platform as a Service - PaaS) - предоставление интегрированной платформы для разработки, тестирования, развертывания и поддержки веб-приложений как сервиса, доступ к которой реализован на основе облачных вычислений.

Сейчас большинство приложений разрабатываются в одной среде, тестируются в другой, а разворачиваются в третьей. Использование концепции PaaS позволяет выполнять все эти этапы разработки программного обеспечения в одной среде, что существенно уменьшает затраты на поддержку отдельных сред для различных этапов [3].

Программное обеспечение, как сервис (Software as a service - SaaS) - это предоставление программного обеспечения через Интернет, когда поставщик разрабатывает веб-приложение и самостоятельно администрирует его. Основным преимуществом использования модели SaaS для пользователя является отсутствие капиталовложений, связанных с установленной, обновлением и поддержкой оборудования и соответствующего программного обеспечения, а также содержанием штата технических работников [4].

Сервис-ориентированная архитектура характеризуется следующими принципами [5]:

- сервисы как компоненты информационной системы являются независимыми от платформы, языка программирования, операционной системы и других особенностей технической реализации. Сервисы взаимодействуют между собой и с другими дополнительными службами, используя открытые стандарты, например, протокол Simple Object Access Protocol (SOAP).

- каждый сервис, входящий в информационную систему, реализует отдельную функцию, которая является логически обособленной и полностью

выполняет какую-то отдельную задачу без необходимости вызова и использования любых других сервисов или программ;

- низкая связность, т.е. сервисы в системах, построенных на основе сервис-ориентированной архитектуры, независимы от других компонентов системы, то есть в них можно вносить изменения, добавлять и удалять и это никак не отразится на других сервисах.

Таким образом, сервисы являются автономными, независимыми от платформы объектами, слабосвязанными между собой.

В общем случае сервис-ориентированная архитектура формируется пользователями, поставщиками сервисов и брокером сервисов (рис. 1). Поставщик сервиса предоставляет информацию о своих ресурсах брокеру сервисов, а пользователи в свою очередь отправляют свои запросы брокеру сервисов, после чего выбирается сервис, который будет предоставлять услуги пользователю.

Одним из главных преимуществ использования парадигмы сервис-ориентированной архитектуры является возможность динамически добавлять и исключать сервисы, поскольку это никак не отразится на функциональности других компонентов системы. В свою очередь, пользователи могут использовать сервисы без знаний о том, каким образом они реализованы и функционируют [6, 7, 8].

Рис. 1. Общая схема сервис-ориентированной архитектуры

Однако, несмотря на переход от классических систем к сервис-ориентированной парадигме, требования для обеспечения качества предоставляемых сервисов в сервис-ориентированных системах (СОС) остаются неизменными. Такие системы должны быть надежными, безопасными, доступными и эффективными в использовании [9]. К сожалению, добиться выполнения всех этих требований для сервис-ориентированных систем сложнее, чем для классических систем, основанных на

непосредственном использовании компьютеров, поскольку сервисы могут предоставляться сторонними разработчиками. Для чего необходимо выявить узкие места, которые являются причиной снижения эффективности работы сервис-ориентированной системы и производительности работы системы в целом.

1. Структурная модель СОС

Введем структурную модель СОС на примере ее частного случая Grid-системы, который позволяет осуществить интеграцию не только локальных, но и географически распределенных информационных ресурсов.

СОС базируется на хорошо прописанных стандартах и сервисах [10], представленных так, что они могут быть обнаружены и задействованные конечным пользователем или приложением. Схематично такую модель можно представить в виде схемы рис. 2.

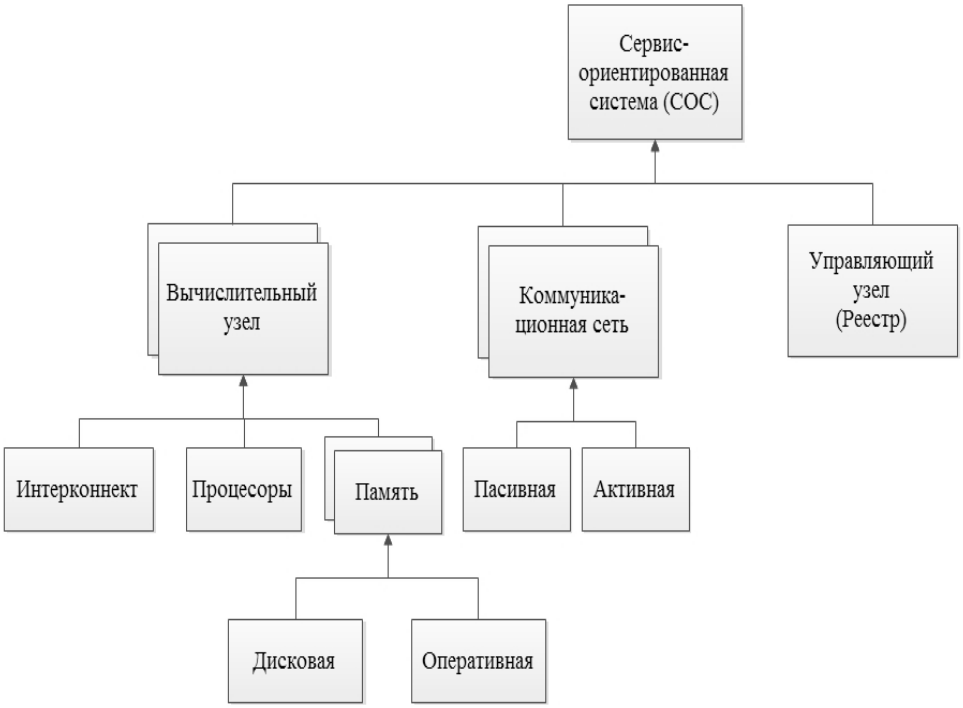

Рис. 2 Структурная модель СОС

Основными компонентами СОС являются:

- *вычислительный узел* — это многопроцессорный компьютер, на котором выполняются сервисы. Задача пользователя может занимать несколько вычислительных узлов, даже кластер целиком. Одновременное выполнение нескольких задач на одном процессоре не допускается.
- *коммуникационная сеть* является ядром информационной сети, обеспечивая некоторые виды обработки данных. Основной задачей является передача данных без ошибок и искажения [11, 10];

- *управляющий* узел характеризуется очередью заданий, которые необходимо распределить между вычислительными узлами, а также представляет собой каталог сервисов, доступных в СОС, так как в него входит реестр, который содержит физическое месторасположение, документацию, версии и срок действия сервисов. Реестр сервисов является одним из основных компонентов блоков архитектуры СОС. Реестр сервисов реализует принцип слабого связывания. Храня месторасположения оконечных точек сервисов, он устраняет тесное связывание, приводящее к жесткой привязке потребителя к провайдеру. Он также облегчает потенциальные сложности замены одной реализации сервиса на другую при необходимости. Реестр сервисов является хорошо масштабируемым; он развивается в соответствии с развитием системы, которую обслуживает. Реестр сервисов позволяет системным аналитикам исследовать корпоративный портфель бизнес-сервисов. Исходя из этого они могут определить, какие сервисы доступны для автоматизации процессов с целью удовлетворения актуальных бизнес-потребностей, а какие нет. Это в свою очередь позволяет узнать, что нужно реализовать и добавить в портфель, формируя каталог доступных сервисов. Реестр сервисов может выполнять функцию управления сервисами, обязывая подписывающиеся сервисы быть согласованными. Реестры сервисов помогают уменьшить время, затрачиваемое на обнаружение информации о сервисе. Без реестра, следящего за сервисами и их взаимоотношениями, СОС-среда не только утрачивает согласованность и контроль, но и приходит в состояние хаоса [12].

Основная идея использования построенной структурной модели СОС состоит в том, чтоб инкапсулировать функциональности крупномодульных приложений в службах, которые распространяются по сети. Если раньше Grid рассматривался как дополнительное средство вычислительных ресурсов, то на данном этапе, с точки зрения Grid-системы [13-16], как частного случая СОС важным является предоставление сервиса для пользователей. Кроме того во многих случаях значительным является управление потоками сервисов, что особенно актуально при использовании Grid для СОС [17-18]. Для решения задач существенной является скорость обмена информации. Таким образом, возникает задача для функционального анализа элементов системы с целью ускорения передачи данных [19].

2. Постановка задачи структурно-функционального анализа сервис-ориентированных систем

Учитывая высокую стоимость создания распределенных систем в которых используются гетерогенные данные и высокопроизводительная техника, к их разработке требуется подходить с позиции системного анализа, в частности метода структурно-функционального анализа.

Рассмотрим общий случай структурно-функционального анализа сервис-ориентированной системы.

Каждый q-й уровень иерархии системы состоит из P_q функциональных элементов. Тогда множество всех функциональных элементов системы можно описать следующим образом:

$$V = \{V_{qp} \mid q = \overline{1, N}, p = \overline{1, P_q}\}, \quad (1)$$

где N — общее количество уровней иерархии; V_{qp} — p-й функциональный элемент q-ого иерархического уровня.

Каждый ФЭ V_{qp} системы характеризуется вектором показателей

$$x_{qp} = (x_{qpj} \mid j = \overline{1, n_{qp}}), \quad (2)$$

где n_{qp} — количество показателей ФЭ V_{qp}, выполняющих набор функций

$$\Phi_{qp} = (f_{qpk} \mid j = \overline{1, m_{qp}}), \quad (3)$$

Каждая из функций f_{qpk} в (3) зависит от значений показателей вектора $\mathbf{x}_{qp}$

$$f_{qpk} = f_{qpk}(\mathrm{x}_{qp}), \quad (4)$$

и влияет на реализацию требований, предъявляемых к ФЭ и системе в целом.

Согласно [20], состав и вид функций (3), (4) определяется в процессе системного анализа.

Пусть

$$X = \{x_{qp}, p = \overline{1, P}, q = \overline{1, N}\}, \quad (5)$$

обобщенное множество показателей. Для простоты изложения откажемся от тройной индексации. Будем считать, что общее число показателей всех ФЭ всех иерархических уровней системы равно N_0. Тогда обобщенный вектор показателей можно представить в виде

$$x = (x_1, x_2, ..., x_{N_0})^T, x \in R^{N_0}, \quad (6)$$

Одной из главных задач структурно-функционального анализа является определение преобразования

$$F : X \to Y, \quad (7)$$

из множества X допустимых показателей системы в пространство Y требуемых свойств по набору количественных требований [19].

При этом требуется решить задачу структурно-параметрической идентификации, позволяющей одновременно определить структуру сервис-ориентированной системы в целом, структуру функциональных элементов всех иерархических уровней и вид преобразования (7).

Проанализируем элемент СОС, включающий хранилище данных, т.е. файловый архив [21]. Для этого исследуем поведение сервис-ориентированной системы в случаях, когда для решения задачи и выполнения вычислений необходимо загрузить из хранилища один или несколько файлов.

3. Структурно-функциональный анализ сервис-ориентированной системы с учетом типов решаемых задач

Рассмотрим пример задачи структурно-функционального анализа для узла сервиса, решающего в СОС задачи определенных типов. Введем множество типов программного кода *K*, а также индикатор типа для кода: если *d* — код (последовательность инструкций), то *type*(*d*) — его тип. Вся задача состоит из частей программного кода различного типа. Экспериментально можно построить функцию эффективности процессора при обработке кода определенного типа w_1: $K \times Arch \to R$, где *Arch* — архитектура процессора, а *R* — множество действительных чисел. Существуют организации, которые предоставляют таблицы значений таких функций для эталонных задач, которые стали общепризнанными стандартами для оценки производительности компьютерных систем. Так, организация производителей компьютеров SPEC (System Performance Evaluation Corporation) [19] публикует результаты исследований производительности процессоров, мультипроцессорных систем, файловых серверов и т.д. При этом в качестве эталонных задач могут использоваться программы, выполняющие только базовые операции (подобные сложению и умножению, например программа Dhrystone), модельные задачи (например, решето Эратосфена и Ханойские башни), ядра (например, программа Linpack) или реальные прикладные задачи (например, модель численного прогнозирования погоды WRF) [21].

Тогда с учетом производительность процессора с частотой f_r, при загрузке *z* для кода типа *k* и архитектуры $arch \in Arch$

$$f_{cpu} = w_1(k, arch) f_r (1 - z), \tag{8}$$

Рассмотрим узел, содержащий *N* процессоров. Пусть производительность интерконнекта (количество информации, передаваемой между процессорами за единицу времени) равна f_{int}. Поскольку обработанную процессором информацию необходимо передавать другим процессорам, при использовании *N* процессоров производительность узла не увеличится ровно в *N* раз. Пусть w_2: $K \times Arch \to R$ — функция, определяющая часть данных, которые необходимо передать между процессорами во время вычислений при выполнении кода единичной (эталонной) длины типа *K*. Таблицу значений этой функции можно построить для каждой задачи, решаемой в системе. Тогда при обработке информации объема *C* необходимо передать информацию объема $Cw_2(k, arch) = C\beta$. Тогда производительность узла (равная количеству обработанных данных) вычисляется по более сложной формуле, чем

$$f_{node}(x_{node}) = \begin{cases} f_{proc}(x_{proc}) f_{mem}(x_{mem}) f_{con}(x_{con}) n_{proc}, \text{для выч.узла,} \\ f_{proc}(x_{proc}) f_{stor}(x_{stor}) f_{con}(x_{con}) n_{proc}, \text{для хранилища,} \end{cases}$$

Количество информации C сначала будет обработано процессором за время C/f_{cpu}, затем эта информация передается другому процессору за время $C\beta / f_{\text{int}}$. Общее время составит $C / f_{cpu} + C\beta / f_{\text{int}}$. Тогда производительность узла с N процессорами составит

$$f_{N\_proc} = \frac{CN}{C / f_{cpu} + C\beta / f_{\text{int}}} = \left(\frac{1}{f_{cpu}} + \frac{\beta}{f_{\text{int}}} \right)^{-1} N \qquad (9)$$

Коэффициент необходимой оперативной памяти $m_1 = m_1(K)$ показывает, сколько оперативной памяти необходимо использовать для вычисления задачи типа K единичной длины. Тогда суммарный объем требуемой оперативной памяти, необходимый для решения задачи, равен $S = \max_{1 \le i \le I}(m_1(type(d_i))size(d_i))$, где d_i - i-ая последовательность инструкций кода, size(d_i) - размер кода d_i. Если такой объем свободной памяти отсутствует, необходимо использовать виртуальную память, т.е. память жесткого диска. Поэтому введем коэффициент требуемой памяти жесткого диска $m_2 = m_2(K)$.

Производительность оперативной памяти считается постоянной f_{RAM}=const. Для вычисления производительности жесткого диска (виртуальной памяти) используем следующий подход. Известно, что производительность жестких дисков существенно зависит от порядка считывания с него информации. В случае последовательного считывания производительность максимальна, а в случае случайного доступа она может уменьшиться в десятки раз. Очевидно, что расположение данных на диске и порядок считывания зависят от типизации кода. Таким образом, производительность жесткого диска представляет собой функцию типа кода, и полезная производительность памяти определяется как $f_{HDD} = f_{HDD}(x_{prod}, k)$ — количество данных, которое может быть записано/считано с жесткого диска, учитывая особенности этих запросов и характеристик конкретного диска [19].

Вычислим полезную производительность узла. Учитывая, что время, потраченное на вычисления, суммируется со временем, потраченным на чтение/запись, получаем, что полезная производительность при обработке кода типа K (сумма обратных величин) определяется выражением

$$f_{node} = \left(\frac{1}{f_{N\_proc}} + \frac{m_1\alpha^1_{RAM} + m_2\alpha^2_{RAM}}{f_{RAM}} + \frac{m_1\alpha^1_{HDD} + m_2\alpha^2_{HDD}}{f_{HDD}} \right)^{-1}, \qquad (10)$$

где коэффициенты $\alpha^{1,2}_{RAM}$ и $\alpha^{1,2}_{HDD}$ показывают, как будет использоваться память. Коэффициент α^1_{HDD} определяет, какая часть данных, которые должны храниться в оперативной памяти, записывается на диск, α^2_{HDD} — какая часть данных будет храниться на жестком диске (как и запланировано), α^1_{RAM} — какая часть данных будет храниться в оперативной памяти (как и запланировано), α^2_{RAM} — какая часть данных, предназначенных для хранения на жестком диске, будет

храниться в оперативной памяти (при использовании кэширования). Таким образом, учитывается возможность использования виртуальной памяти и кэширования. При этом $\alpha^1_{RAM} + \alpha^1_{HDD} = 1$ и $\alpha^2_{RAM} + \alpha^2_{HDD} = 1$.

4. Результаты численного моделирования

Пример 1.Оценка эффективности передачи одного файла данных в сервис-ориентированной системе

Проиллюстрируем применение предложенного подхода для оценки производительности сервис-ориентированной системы при выполнении заданий разного типа.

Общая постановка задачи сводится к следующему.

Задана общая архитектура вычислительного узла (рис. 3) и типы выполняемых заданий. Требуется на основе предложенного подхода к структурно-функциональному моделированию СОС определить следующие характеристики:

1) производительность СОС при выполнении однотипных заданий;

2) выявить слабые (критичные) места СОС (т.е. те места, которые ограничивают суммарную производительность системы).

Рассмотрим систему, которая обладает следующими свойствами.

1. Система, состоящая из 3 однопроцессорных компьютеров и 2 кластеров, при этом один из однопроцессорных компьютеров используется в качестве управляющего узла, обеспечивающего только поступление в сегмент пользовательских задач.

2. Взаимодействие между узлами осуществляется с помощью 2 маршрутизаторов.

3. Пропускная способность сети составляет 1000 Мбит/с=125 Мбайт/с.

4. Для простоты будем считать, что производительность процессоров в формуле (8) пропорциональна только тактовой частоте f_r и не зависит от типа выполняемого кода (код считаем однородным).

5. Будем считать, что в рассматриваемом примере однопроцессорные компьютеры имеют частоту f_r=1 ГГц.

6. Оперативная память однопроцессорного компьютера составляет 256 Мбайт.

7. Жесткий диск такого компьютера имеет размер 60 Гбайт.

8. Каждый вычислительный узел содержит 16 процессоров с частотой 2,5 ГГц. Объем локальной оперативной памяти вычислительного узла составляет 128 Мбайт на каждый процессор и емкость каждого из 16 жестких дисков — 80 Гбайт.

Предположим, что для решения каждой задачи перед выполнением вычислений необходимо передать из хранилища один файл размером в 500 Мб.

Требуется. Оценить производительность сервис-ориентированной системы.

Для решения данной проблемы изначально, вычислим производительность узла.

Учитывая, что время, потраченное на вычисления, суммируется со временем, потраченным на чтение/запись, тогда полезная производительность определяется выражением [19] при наличии необходимого объема данных, используя формулу 10 вычислим производительность кластера в целом составляет:

$$f_{node} = \left(\frac{1}{8{,}33 \cdot 10^9} + \frac{0{,}1 \cdot 1}{10^9} \right)^{-1} 16 = 7{,}27 \cdot 10^{10} \, байт / с.$$

Для персонального компьютера эта величина составляет:

$$f_{node} = \left(\frac{1}{4 \cdot 10^9} + \frac{0{,}2 \cdot 1 + 0 \cdot \alpha_{RAM}^2}{10^{10}} + 0 \right)^{-1} = (2{,}5 \cdot 10^{-10} + 2 \cdot 10^{-11})^{-1} \approx 3{,}70 \cdot 10^9 \, байт / с.$$

Пусть в среднем на персональные компьютеры каждую секунду подается x_1 и x_2 задач. На кластеры соответственно подается y_1 и y_2 задач за секунду. Тогда для обеспечения выполнения этих задач на маршрутизатор, показанный на рис. 3 слева каждую секунду нужно загружать из хранилища $5 \cdot 10^8 (x_1 + x_2)$ байт, а на маршрутизатор справа — $5 \cdot 10^8 (y_1 + y_2)$ байт. Кроме загрузки данных для узлов, необходимо загружать задачи непосредственно на вычислительные элементы.

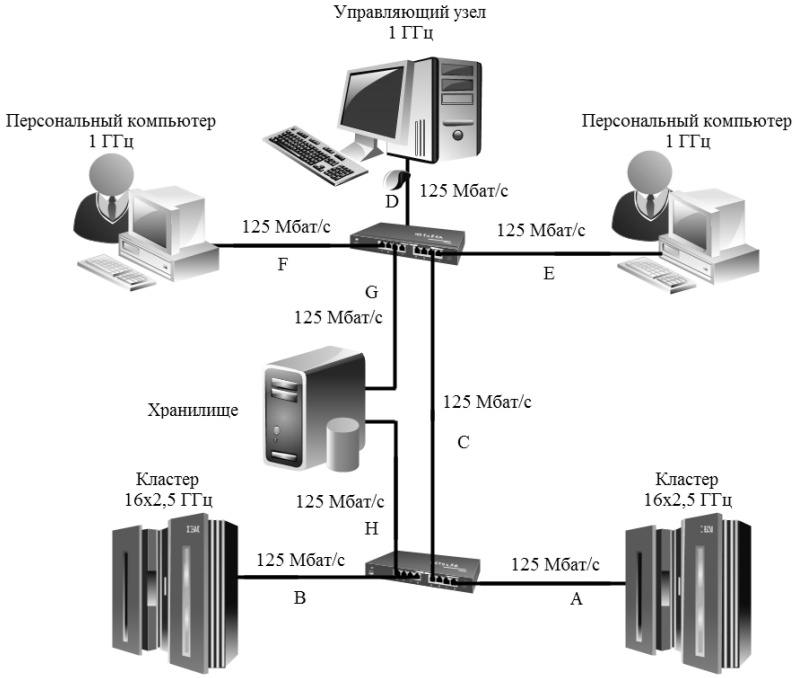

Рис. 3 Сетевая топология СОС с хранилищем

Соответственно, дополнительно каждую секунду необходимо передавать: $10^6 y_1$ на участке A, $10^6 y_2$ на участке B, $10^6 (y_1 + y_2)$ на участке C, $10^6 x_1$ на участке E, $10^6 x_2$ на участке F и $10^6 (x_1 + x_2 + y_1 + y_2)$ на участке D.

Таким образом, общая нагрузка на сеть составляет: $(10^6+5\cdot10^8)y_1$ на участке A, $(10^6+5\cdot10^8)y_2$ на участке B, $10^6(y_1+y_2)$ на участке C, $10^6(x_1+x_2+y_1+y_2)$ на участке D, $(10^6+5\cdot10^8)x_1$ на участке E, $(10^6+5\cdot10^8)y_2$ на участке F, $5\cdot10^8(x_1+x_2)$ на участке G, $5\cdot10^8(y_1+y_2)$ на участке H.

При этом на величины x_1 и x_2 накладываются ограничения: их значения не могут превышать количество задач, которое можно выполнить на одном персональном компьютере за одну секунду $x_{1,2}\le 3{,}70\cdot10^9/10^9=3{,}7$. Для кластеров такое ограничение имеет вид: $y_{1,2}\le 7{,}27\cdot10^{10}/10^9=72{,}7$.

Таким образом, поставленная задача свелась к следующей задаче линейного программирования.

Необходимо максимизировать функцию $x_1+x_2+y_1+y_2$, при следующих ограничениях на x_1, x_2, y_1, y_2:

$$(10^6+5\cdot10^8)y_1\le sp(A),$$
$$(10^6+5\cdot10^8)y_2\le sp(B),$$
$$10^6(y_1+y_2)\le sp(C),$$
$$10^6(x_1+x_2+y_1+y_2)\le sp(D),$$
$$(10^6+5\cdot10^8)x_1\le sp(E),$$
$$(10^6+5\cdot10^8)x_2\le sp(F),$$
$$5\cdot10^8(x_1+x_2)\le sp(G),$$
$$5\cdot10^8(y_1+y_2)\le sp(H),$$
$$x_1\le 3{,}7,$$
$$x_2\le 3{,}7,$$
$$y_1\le 72{,}7,$$
$$y_2\le 72{,}7,$$
$$x_1\ge 0, x_2\ge 0, y_1\ge 0, y_2\ge 0.$$

Пусть пропускная способность сети на всех участках оставляет 1000 Мбит/с=125 Мбайт/с. рис. 3, имеем $x_1+x_2=y_1+y_2=0{,}25$. Тогда, производительность СОС составляет $x_1+x_2+y_1+y_2=0{,}25$ задач/с. При этом столь низкая производительность объясняется большим объемом данных, которые необходимо пересылать по сети для выполнения одной задачи [19].

Пример 2. Оценка эффективности передачи несколько файлов данных меньшего размера в сервис-ориентированной системе

Рассмотрим теперь случай, когда для выполнения задач нужно загрузить из хранилища не один файл большого объема, а несколько файлов меньшего размера. Предположим, что в результате выполнения задачи на

вычислительном узле генерируются файлы, которые необходимо загрузить в хранилище. Таким образом, внесем следующие коррективы:

1. Размер файлов, которые передаются по сети, составляют 1 Мбайт. При этом размер выполняемого кода (количество необходимых для выполнения машинных инструкций) составляет 1000 Мбайт.

2. Для выполнения задач на вычислительный узел необходимо загрузить из хранилища 10 файлов размером в 25 Мб каждый (всего 250 Мб).

3. После выполнения задания полученный результат необходимо сохранить в хранилище в виде одного файла размером 250 Мб.

4. Скорость чтения из хранилища при считывании 1, 2, 3, ... файлов составляет соответственно 450, …, 200 (десятый член последовательности), … Мбайт/с.

5. Скорость записи в хранилище, при записи 1, 2, 3, ... файлов составляет соответственно 125, 110, 100, 95, ... Мб/с.

Требуется. Определить производительность хранилища при чтении из него 10 файлов небольшого размера.

Так как в этом случае речь идет о чтении 10 файлов, то производительность жесткого диска (или дисков) хранилища составляет 200 Мб/с: $f_{HDD} = 2 \cdot 10^8$ (байт/с), имеем $f_{read} = 2 \cdot 10^8$ (байт/с).

Аналогично найдем производительность хранилища при загрузке файла большого размера. В этом случае (т.е. при записи одного файла) производительность диска хранилища составляет 125 Мб/с или $f_{HDD} = 1{,}25 \cdot 10^8$ (байт/с). Т.е. $f_{write} = 1{,}25 \cdot 10^8$ (байт/с).

Таким образом, производительность хранилища при чтении является достаточной для обеспечения регулярного чтения (выгрузки в сеть) 200 Мб/с и записи (загрузки из сети) только 125 Мб/с. В обоих случаях это не превышает пропускной способности соединения хранилища и сети (2 соединения по 125 Мб/с каждое).

Пусть в среднем на персональные компьютеры каждую секунду подается x_1 и x_2 задач. На кластеры соответственно подается y_1 и y_2 задач за секунду. Тогда для обеспечения выполнения этих задач каждую секунду необходимо загружать из хранилища $2{,}5 \cdot 10^8 (x_1 + x_2)$ байт на маршрутизатор, показанный на рис. 3 слева, и $2{,}5 \cdot 10^8 (y_1 + y_2)$ на маршрутизатор справа. В результате решения задачи от маршрутизатора слева поступает $2{,}5 \cdot 10^8 (x_1 + x_2)$ байт, а от маршрутизатора справа — $2{,}5 \cdot 10^8 (y_1 + y_2)$ байт.

Остальные ограничения можно получить аналогично предыдущему примеру. При этом необходимо учесть следующее: хранилище должно успевать считывать и записывать информацию с заданной скоростью. Вычислим эту величину. При выполнении одной задачи хранилище выгружает файлы размером 250 Мб со скоростью 200 Мб/с и загружает файл размером 250 Мб со скоростью 120 Мб/с. На выгрузку файлов уходит $T_{out} = 250 / 200 = 1{,}25$ секунд, на

загрузку $T_{in} = 250/125 = 2$ секунд, что в сумме составляет 3,25 сек. Для обеспечения выполнения $x_1 + x_2 + y_1 + y_2$ задач, хранилище будет занято в течение $3{,}25 \cdot (x_1 + x_2 + y_1 + y_2)$ с, что должно быть меньше 1 с.

Поставленная задача свелась к следующей задаче линейного программирования:

Необходимо максимизировать функцию $x_1 + x_2 + y_1 + y_2$, при следующих ограничениях на x_1, x_2, y_1, y_2:

$$(10^6 + 5 \cdot 10^8) y_1 \le sp(A),$$
$$(10^6 + 5 \cdot 10^8) y_2 \le sp(B),$$
$$10^6 (y_1 + y_2) \le sp(C),$$
$$10^6 (x_1 + x_2 + y_1 + y_2) \le sp(D),$$
$$(10^6 + 5 \cdot 10^8) x_1 \le sp(E),$$
$$(10^6 + 5 \cdot 10^8) x_2 \le sp(F),$$
$$5 \cdot 10^8 (x_1 + x_2) \le sp(G),$$
$$5 \cdot 10^8 (y_1 + y_2) \le sp(H),$$
$$x_1 \le 3{,}7,$$
$$x_2 \le 3{,}7,$$
$$y_1 \le 72{,}7,$$
$$y_2 \le 72{,}7,$$
$$3{,}25(x_1 + x_2 + y_1 + y_2) \le 1$$
$$x_1 \ge 0, x_2 \ge 0, y_1 \ge 0, y_2 \ge 0.$$

Пропускная способность сети составляет 1000 Мбит/с=125 Мбайт/с, скорость соединения на всех участках сети составляет $1{,}25 \cdot 10^8$ (байт/с). Тогда, учитывая предположения, что жесткий диск такого компьютера имеет размер 60 Гбайт, а каждый многопроцессорный вычислительный узел содержит 16 процессоров с частотой 2,5 ГГц. Размер общей оперативной памяти составляет 2 Гбайт и емкость общего жесткого диска — 500 Гбайт, после чего получим следующее решение $x_1 + x_2 \le 0{,}25$, $y_1 + y_2 \le 0{,}25$. Однако вследствие предположения что производительность памяти (т.е. скорость чтения/записи) составляет 10 Гбайт/с, производительность составляет $x_1 + x_2 + y_1 + y_2 = \frac{4}{13}$ задач/с. Такая небольшая производительность объясняется большими объемами данных, которые передаются по сети для выполнения задач, и малой производительностью хранилища [19].

Выводы

Проведение исследования для функциональных элементов всех иерархических уровней позволило сформулировать прямые и обратные задач

оптимизации СОС и ее отдельных компонентов. Решение прямых задач обеспечивает максимизацию производительности системы с учетом типов решаемых задач [19]. Проанализированы результаты экспериментальных исследований для оценки эффективности СОС разной аппаратной конфигурации и различных функциональных элементов. В том числе рассчитаны задачи определения эффективности передачи одного и нескольких файлов в реализации сервис-ориентированной системы.

После чего можно сделать вывод, что наиболее узким местом сервис-ориентированной системы при решении задач, требующей передачи большого количества файлов небольшого объема является хранилище. Т.е. для обеспечения эффективной работы СОС, необходимо модернизировать скорость передачи данных внутри хранилища, что является перспективой для дальнейшей работы и исследований. Разработанная архитектура может использоваться для решения прикладных задач агромониторинга [22, 23].

Литература

1. Kussul N., Kussul O., Skakun S. Probabilistic estimation of trust model and threat resistance analysis in service-oriented systems // Int. J. on Information Models and Analyses. – 2012, no 1. – P. 28-46.
2. Cloud Computing: Distributed Internet Computing for IT and Scientific Research / M.D. Dikaiakos, D. Katsaros, P. Mehra [et al.] // Internet Computing. — 2009. — V.13, Issue 5. – P. 10-13.
3. Елманова Е. Коротко о вычислениях в «облаке» / Е. Елманова // КомпьютерПресс. — 2010. — №3. — С.102-104.
4. Черняк Л. SaaS - конец начала [Електронний ресурс] / Леонид Черняк // Открытые системы. — 2007. — №10. — Режим доступу до журн.: http://www.osp.ru/os/2007/10/4706040/
5. Service-Oriented Computing: State of the Art and Research Challenges / Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar [et al.] // IEEE Computer. — 2007. — №11. — V.40. —P. 38-45.
6. Alonso G. Web Services: Concepts, Architectures and Applications / Gustavo Alonso. – Berlin Heidelberg: Springer-Verlag. —2004. —359 p.
7. Erl T. Service-oriented architecture: a field guide to integrating XML and Web services / Thomas Erl. — Prentice Hall PTR. — 2004. — 536 p.
8. Hurwitz J. Service Oriented Architecture For Dummies / J.Hurwitz, C.Baroudi, R.Bloor and M.Kaufman. — UK: Wiley & Sons. — 2006. — 384 p.
9. Dong J. High-Assurance Service-Oriented Architectures / Jing Dong, Raymond A. Paul, Liang-Jie Zhang // IEEE Computer. — 2008. — №8, V.41. —P. 27-28.
10. Павлов А.А. Информационные технологии и алгоритмизация в управлении / А.А.Павлов, С.Ф.Теленик. — К.: Техника. — 2002. С. — 344 с.
11. Kussul N. Grid and sensor web technologies for environmental monitoring / N. Kussul, A. Shelestov, S. Skakun // Earth Sci. Inf. — 2009. — №2(1-2). — P. 37–51.

26. Kussul, N. and Shelestov, A. and Skakun, S. Grid and sensor web technologies for environmental monitoring // Earth Science Informatics. — 2009. — vol. 2, no. 1-2. — P. 37-51.

12. OASIS. *UDDI* – [Электронный ресурс] –http://uddi.xml.org (2012).

13. Shelestov, A.Yu. and Kussul, N.N. and Skakun, S.V. Grid technologies in monotoring systems based on satellite data // Journal of Automation and Information Sciences. — 2006. — vol. 38, no. 3. — P. 69-80.

14. Bakan, G.M. and Kussul. Fuzzy ellipsoidal filtering algorithm of static object state // Problemy Upravleniya I Informatiki (Avtomatika). — 1996. — no. 5. — P. 77-92.

15. Lecca, G. and Petitdidier, M. and Hluchy, L. and Ivanovic, M. and Kussul, N. and Ray, N. and Thieron, V. Grid computing technology for hydrological applications // Journal of Hydrology. — 2011. — vol. 403, no. 1-2. — P. 186-199.

16. Kravchenko, A.N. and Kussul, N.N. and Lupian, E.A. and Savorsky, V.P. and Hluchy, L. and Shelestov, A.Yu. Water resource quality monitoring using heterogeneous data and high-performance computations // Cybernetics and Systems Analysis. — 2008. — vol. 44, no. 4. — P. 616-624.

17. Shelestov, A.Yu. and Kussul, N.N. Using the fuzzy-ellipsoid method for robust estimation of the state of a grid system node // Cybernetics and Systems Analysis. — 2008. — vol. 44, no. 6. — P. 847-854.

18. Kussul, N. and Skakun, S. Neural network approach for user activity monitoring in computer networks // IEEE International Conference on Neural Networks - Conference Proceedings. — 2004. — vol. 2. — P. 1557-1561.

19. Куссуль Н.Н. Grid-системы для задач исследования Земли. Архитектура, модели и технологи/ Н.Н. Куссуль, А.Ю. Шелестов // — К.: “Наукова думка”. — 2008. С. – 452 с.

20. Згуровський М. З. Системний аналіз: проблеми, методологія, застосування / М. З. Згуровський, Н. Д. Панкратова. – К. : Наук. думка, 2005. – 744 с.

21. Мордвинов В. И. Численные методы анализа и прогноза погоды : учеб. пособие / В. И. Мордвинов. — М. : Изд-во Иркут. гос. ун-та, 2008. — 143 с.

22. Kussul, N, Skakun, S, Shelestov, A, Kravchenko, O, Gallego, J & Kussul, O. Crop area estimation in Ukraine using satellite data within the MARS project // IEEE International Geoscience and Remote Sensing Symposium, IEEE, Munich, Germany. — 2012. — P. 3756-3759.

23. Gallego, J. and Kravchenko, A.N. and Kussul, N.N. and Skakun, S.V. and Shelestov, A.Yu. Efficiency assessment of different approaches to crop classification based on satellite and ground observations // Journal of Automation and Information Sciences . — 2012. — vol. 44, no. 5. — P. 67-80.