ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 9**, №1(36), с. 85–108

УДК 004.75:004.052.3:004.042

А. А. Талалаев, В. П. Фраленко

Отказоустойчивая система организации высокопроизводительных вычислений для решения задач обработки потоков данных

Аннотация. В работе представлен обзор существующих систем организации отказоустойчивых вычислений; рассмотрены функциональные характеристики разработанной высокопроизводительной системы на основе вычислительного ядра, специального интерфейса и прикладных модулей. В основе предлагаемой системы организации вычислений лежит распределенная NoSQL СУБД Apache Cassandra, обеспечивающая механизмы отказоустойчивого хранения и автоматической репликации данных в гетерогенной вычислительной среде. Система, оснащенная специальным графическим интерфейсом, позволяет разрабатывать решения для различных прикладных областей. Подключаемые модули могут выполнять в том числе и функции визуализации потоков данных.

Ключевые слова и фразы: организация вычислений, отказоустойчивость, база данных, графический интерфейс, модуль, визуализация, гетерогенная среда.

Введение

Организационно-технические детали реализации программного обеспечения существенно влияют на его дальнейшие эксплуатационные характеристики. В частности, при выработке мер достижения высокой доступности информационных сервисов рекомендуется опираться на следующие архитектурные принципы:

- апробированность всех процессов и составных частей;
- унификация процессов и составных частей;

Исследование выполнено при частичной финансовой поддержке РФФИ (проекты № 15–29–06945–офи_м, 16–29–12839–офи_м, 17–29–07002–офи_м и 18–07–00014–а).

10.25209/2079-3316-2018-9-1-85-108

- управляемость процессов, контроль состояния частей;
- автоматизация процессов;
- модульность архитектуры;
- ориентация на простоту решений.

Наиболее критичные для информационных систем случаи — отказы целых вычислительных узлов, что приводит к необходимости переконфигурирования вычислительной сети. Далее предлагается обзор решений, используемых для обеспечения отказоустойчивости вычислительных процессов.

В системе «GRACE» [1] в случае отказа одного из узлов все вычисления, отправленные на него с других компьютеров, будут автоматически переданы на вычисление другим свободным узлам. Не менее важным является реагирование на появление дополнительных ресурсов во время вычислений. В «GRACE» все вычислительные ресурсы образуют иерархию ресурсов, для каждого уровня которой динамически вычисляется доступная в ней свободная вычислительная мощность. В случае появления (подключения) нового компьютера происходит динамическая реконфигурация иерархии вычислительных ресурсов, алгоритм внешнего планирования начинает учитывать вновь появившиеся узлы.

Одним из возможных подходов к повышению отказоустойчивости является дублирование вычислений и важных вспомогательных функций, например, функций хранения данных; гранулярный характер решаемых задач может позволять вводить дублирование вычислений для того, чтобы одни и те же задачи выполнялись на нескольких независимых узлах [2]. Другой подход позволяет обеспечить отказоустойчивость за счет портальной работы с данными: в работе [3] задача разбивается на подзадачи, между которыми организуются каналы связи. Порталы используются для фиксации переходов Т-подзадач из одного вычислительного пространства в другое. В случае потери связи с узлом или выхода последнего из строя портал содержит именно те Т-функции, что необходимо вычислить заново. Планировщик старается пересылать промежуточные функции на промежуточные узлы, а на роль последних выдвигает наиболее надежные. Тем не менее, и промежуточные узлы могут выйти из строя. Если выведен из строя промежуточный узел квази-иерархии, то его подчиненные узлы пытаются найти свое место у следующего вышестоящего лидера и т.д.

В работе [4] рассмотрено динамическое распределение запросов на использование функциональных ресурсов, рассредоточенных по узлам вычислительной системы, определены рациональные по производительности и надежности варианты размещения этих ресурсов. Проведена оценка отказоустойчивости и производительности сравниваемых конфигураций при реализации динамического распределения запросов через канал распределенной вычислительной сети, когда в процессе решения задачи могут формироваться запросы, каждый из которых требует доступ к нескольким ресурсам разного типа.

Исследование [5] рассматривает отказоустойчивую многопроцессорную систему с непрерывным взаимным контролем работоспособности вычислительных модулей. На основе принципов характеризационного анализа получены необходимые и достаточные условия, при выполнении которых для заданной степени отказоустойчивости структура межмодульных диагностических связей обеспечивает локализацию всего множества неисправных вычислительных модулей.

В работе [6] введено понятие d-ограниченной компоненты связности графа вычислительной системы. Исследованы функции отказоустойчивости кольцевой и полносвязной структур системы, ограничивающие область определения этой функции для структур, промежуточных в отношении их степени. Постановка задачи анализа структурной отказоустойчивости вычислительной системы, представленная в работе, впервые рассматривает в качестве критических параметров работоспособности минимально допустимый размер компоненты связности графа вычислительной системы и ее предельный диаметр. Дано определение структурно отказоустойчивой системы, проведено исследование полярных в отношении связности кольцевой и полносвязной структур.

В работе [7] предложена следующая архитектура. Для обнаружения отказов ресурса и дальнейшего восстановления совместно работают диспетчер восстановления и мониторы ресурсов. Мониторы ресурсов следят за состоянием ресурсов, периодически опрашивая ресурсы с использованием библиотек ресурсов.

Опрос проводится в два этапа: коротким запросом «LooksAlive» и более долгим и детальным запросом «IsAlive». Когда монитор ресурсов обнаруживает отказ ресурса, он извещает об этом диспетчер восстановления и продолжает следить за ресурсом. Диспетчер восстановления поддерживает ресурсы и рабочее состояние групп ресурсов. Он также отвечает за выполнение восстановления, когда ресурс отказывает, и

вызывает монитор ресурсов в ответ на действия пользователя или на отказы. После обнаружения отказа ресурса диспетчер выполняет действия по восстановлению, которые включают либо перезапуск ресурса и зависящих от него ресурсов, либо перемещение целой группы ресурсов на другой узел. В процессе восстановления после отказа группа рассматривается как единое целое, чтобы зависимости ресурсов были правильно восстановлены.

В публикации [8] предлагается в вектор состояния отдельных узлов добавить «маску активности» узлов, формируемую локально на каждом узле по факту получения вектора состояния от других узлов системы. Таким образом, кроме сигнала извещения об активности, формируемого и отправляемого самим узлом, контролируется физическая доступность данного узла и, соответственно, физическая целостность канала связи. Функции запуска, останова, контроля состояния задач, запускаемых на конкретном узле распределенной системы управления, возлагаются на данный узел. Однако, для успешного выполнения этих функций следует учитывать состояние системы в целом, что в децентрализованной системе приводит к необходимости выделения функции арбитра системы.

На основании текущей информации о состоянии системы арбитр дает команды узлам на запуск и останов процессов. Функция арбитра является переходящей. Для каждой задачи в конфигурации системы описывается, на каких узлах задача может запускаться, в какой последовательности, также фиксируется количество одновременно запущенных в системе экземпляров данной задачи. При первоначальном запуске задачи арбитр подает соответствующую команду на первый активный узел из очереди приоритетов для данной задачи, на последующие активные узлы из очереди — для запуска остальных экземпляров задачи. В случае пропадания одного из узлов из системы выдается команда на запуск задачи на следующем по приоритету активном узле, на котором данная задача еще не запущена. При восстановлении активности выведенного из системы узла арбитр выдает команду на останов всех экземпляров задачи, а затем — на запуск с учетом изменений в очереди узлов, т.е. со сдвигом к ее началу, что обеспечивает обратную миграцию задач на восстановленный узел.

В исследовании [9] рассматриваются вопросы, связанные с проведением расчетов в распределенных вычислительных системах, компоненты

которых подвержены отказам. В работе приводятся: определения системы, сбоя, ошибки, отказа и модели сбоя; наиболее важные результаты исследований отказов в параллельных вычислительных системах, в том числе с большими группами дисков; основные существующие методы восстановления и распространенные программные реализации обеспечения отказоустойчивости. Развивается подход обеспечения отказоустойчивости на уровне пользователя. Данный подход требует непосредственного участия разработчика прикладной программы в реализации метода обеспечения отказоустойчивости, в частности в формировании контрольных точек и процедур восстановления. Предложена схема сохранения в памяти вычислительных узлов данных прикладной программы, формирующих согласованную глобальную контрольную точку. В ее рамках осуществляется дублирование локальных контрольных точек, что позволяет восстановить вычислительный процесс, если число отказов не превосходит допустимого для данной схемы уровня.

В рамках C++-библиотеки «T-Sim» создан отдельный тип переменных — неготовые переменные с заданным временем жизни [**10**]. Этот тип обладает всеми свойствами неготовой переменной, для него определена сериализация, что позволяет корректно оперировать переменными данного типа в распределенной памяти. Но, в отличие от обычной неготовой переменной, в шаблонных параметрах данного типа можно указать еще и время жизни такой переменной. Обычная неготовая переменная имеет два состояния: «готова» (и тогда значением данной переменной является вычисленное значение) и «не готова», неготовая переменная со временем жизни имеет третье, дополнительное состояние — результат до сих пор не вычислен, а время жизни истекло. Недостатком подхода является сложность применения в случае неравновесных гранул параллелизма, которые, к примеру, могут быть порождены при рекурсивном вычислении.

В случае перехвата отказа одного из вычислительных узлов все оставшиеся активными узлы удаляют адрес данного узла из списка доступных для проведения вычисления и выполняют повторную посылку тех задач, которые были на него отправлены. Такой подход позволяет обеспечить отказоустойчивость даже для задач с неравновесными гранулами параллелизма, к тому же использование данного подхода позволяет довести до конца обработку задач, которые были направлены на отказавший узел, а во время вычисления на узле

случился сбой, т.к. после того, как сбой будет обнаружен, задача будет заново перенаправлена на другой узел. Недостатком такого подхода является наличие высоких накладных расходов на сохранение задач и требуемой оперативной памяти на поддержание списка задач. Достоинством является движение в сторону локальной синхронизации, т.к. каждый узел помнит те задачи, для которых было проведено локальное назначение.

В диссертационной работе [**11**] разработаны модели формирования оптимальных модульных систем на основе мультиверсионной программной архитектуры, обеспечивающие распределение мультиверсионных процедур по модулям программной системы с учетом заданных требований и достижение требуемой надежности системы. Разработана и программно реализована среда исполнения на базе компонентной архитектуры для распределенной мультиверсионной системы, которая позволяет подключать к среде исполнения любое количество программных модулей, распределяя вычислительную нагрузку на множество машин.

В диссертационной работе [**12**] создан метод оперативного переразмещения программ в отказоустойчивых мультикомпьютерных системах, использующий диагональное распределение скользящего резерва непосредственно в матрице процессоров. Метод позволяет минимизировать время межпроцессорного обмена данными путем целенаправленного пошагового снижения отклонения указанного времени от нижней оценки наибольшей частной коммуникационной задержки, определяемой исходя из длин заранее формируемых (статических) маршрутов передачи данных в присутствии неоднородностей, обусловленных отказами физической структуры системы. Разработан аппаратно-ориентированный алгоритм, в рамках которого выполняется поиск резервного модуля для замещения отказавшего процессора на множестве ближайших к отказу резервных модулей. Алгоритм позволяет снизить время поиска нового варианта размещения программ в системе после возникновения отказа.

В исследовании [**13**] предлагается асинхронная отказоустойчивая распределенная система, которая на основе собираемой статистики и требований пользователя принимает решения об использовании той или иной оптимизации. Кроме того, собираемая информация позволяет системе подбирать оптимальные параметры используемых алгоритмов. Проблема миграции задач решается следующим образом. Вводится

дополнительная системная служба управления задачами. Эта служба хранит информацию об идентификаторе задачи и соответствующей ей группе. При изменении состава группы эта информация обновляется. Если некоторый клиент после длительного отсутствия коммуникаций теряет связь с группой, вследствие ее полного или частичного изменения, он может обратиться с запросом к этой службе и получить адреса новых членов этой группы. Естественно, возникает проблема с поиском самой этой службы и ее миграцией. Для этого каждый вычислительный узел хранит идентификаторы физических процессоров, входящих в состав виртуального процессора, на котором работает служба [14].

Следует упомянуть об универсальном программном обеспечении для повышения отказоустойчивости грид-систем «HPC4U» [15, 16]. Целью группы разработчиков является расширение потенциала использования грид-систем для решения сложных задач путем разработки компонентов программного обеспечения, реализующих надежную и достоверную среду выполнения грид-приложений, и увязки этого с соглашениями сервисного уровня (Service Level Agreements, SLA) и с промышленными кластерами. Программное обеспечение позволяет прозрачным образом добавлять в приложения для кластеров точки синхронизации, выполнять миграцию задач.

В цикле работ о системе «Dryad» [17–20] рассказывается о среде исполнения распределенных приложений, которая берет на себя следующие функции: планирование и управление распределенными заданиями; управление ресурсами; обеспечение отказоустойчивости; мониторинг. Задание в «Dryad» представляет собой направленный ациклический граф, где вершины представляют собой программы, а ребра графа — каналы данных. Этот логический граф отражается исполняемой средой на физические ресурсы, находящиеся в кластере. В общем случае количество вершин в графе превышает количество физических вычислительных узлов в кластере.

«HTCondor» [21] – специализированная система управления нагрузкой на вычислители. Поддерживает работу с очередями, приоритезацию задач, политики планирования, мониторинг и управление ресурсами. При недоступности удаленной машины осуществляется миграция задач на другие вычислители, это выполняется за счет использования механизма контрольных точек. В Linux-системах для их создания используется модуль «BLCR», позволяющий осуществлять сохранение полного контекста процесса и его дальнейший перезапуск. «HTCondor»

и «BLCR» в настоящее время не имеют поддержки режима отказоустойчивости для программ с регулярными межпроцессными обменами данных, что существенно ограничивает применимость такого подхода. Однако, использование библиотек, подобных «NR-MPI» [22], позволяет обойти данное ограничение. «NR-MPI» имеет набор дружественных программисту программных интерфейсов резервного копирования. Проведенные эксперименты показывают, что библиотека позволяет продолжить вычислительный процесс без перезапуска, при этом накладные расходы оказываются небольшими даже при использовании десятков тысяч вычислительных ядер.

Функция динамического переконфигурирования вычислительной сети крайне востребована в ситуации, когда в счете принимают участие десятки, сотни и даже тысячи компьютеров. Программное обеспечение, предназначенное для работы над сложными математическими и прикладными задачами, требует использования огромного числа вычислительных устройств, которые могут непредсказуемо выходить из строя. Далее будет описано решение, ориентированное на крупногранулярный параллелизм задач и коренным образом отличающееся от ранее известных подходов.

1. Внутреннее устройство вычислительного ядра

В основе предлагаемой системы организации вычислений лежит распределенная NoSQL СУБД Apache Cassandra [23], обеспечивающая механизмы отказоустойчивого хранения и автоматической репликации данных, что позволяет обеспечить отказоустойчивый режим работы при решении прикладных задач. Общая архитектура представлена на рис. 1. Все данные, обрабатываемые в ходе решения прикладной задачи (задач), пользователя хранятся в СУБД, также, как и внутренние состояния модулей, вычислителей и подсистемы диспетчеризации. Фактически, СУБД используется не только как хранилище данных, но и как «очередь сообщений», обеспечивая реализацию механизмов информационного обмена и взаимодействие между отдельными элементами системы. Благодаря использованию подобного подхода повышается отказоустойчивость хранения данных и, соответственно, надежность вычислительной среды.

Опишем элементы программной системы и принципы их функционирования. Основными компонентами разработанной системы

Рис. 1. Общая архитектура системы

распределенных вычислений являются ресурсы, программные модули обработки данных, вычислитель, диспетчер распределения нагрузки, файлы описания прикладной задачи, база данных в СУБД. Одной из основных сущностей системы является понятие ресурса. Ресурс (Resource) — это данные, обрабатываемые в ходе решения прикладной задачи пользователя. Формально, ресурс представляет собой пару <идентификатор, значение>, при этом «идентификатор» — уникальный ключ, а «значение» — массив байт, хранящий произвольные данные в сериализованном виде. Подобное представление соответствует отображению данных в NoSQL-системах, хранящих данные в виде пар «ключ-значение».

Явная типизация данных отсутствует, задача их корректной интерпретации ложится на разработчика модулей обработки данных. Программисту предлагается использовать структуры-дескрипторы. Дескриптор хранит список уникальных имен (полей ресурса) и соответствующие им идентификаторы, являющиеся ключами для доступа к пользовательским данным, сохраненным в базе данных (БД). Иерархическая структура ресурса, описываемая с использованием дескриптора, может быть изменена в ходе обработки данных модулями, могут быть добавлены новые поля или удалены существующие, что также позволяет частично обойти ограничение Apache Cassandra на максимальный размер одного экземпляра пользовательских данных (два гигабайта).

Важным элементом разработанной системы является понятие модуля обработки данных. Модуль (Plugin) — это программная реализация некоторой группы функции обработки данных (ресурсов). Для каждого модуля определен «контекст вызова» и дополнительная метаинформация, используемая в системе диспетчеризации нагрузки для управления процессом вычислений. Контекст вызова представлен структурой, аналогичной описанному ранее ресурсу и дополнительной метаинформацией, представленной в виде описания модуля в формате xml. В процессе работы модуля контекст может изменяться, там, к примеру, могут быть сохранены важные данные, для восстановления которых требуется длительное время. Контексты сохраняются в базе данных в транзакционном режиме, таким образом при сбое уже не придется повторно вычислять эти данные. Модули могут осуществлять графическую визуализацию тех или иных данных, для вызова таких программных функций разработан специальный интерфейс программирования.

Каждый модуль имеет множество каналов передачи данных, посредством которых он может получать или отсылать данные, взаимодействуя с другими модулями. Понятие канала (Channel) — удобная абстракция, обеспечивающая идентификацию поступающих на обработку данных, что упрощает программную реализацию модулей.

Для модулей, использующих GPGPU-технологии, ядро системы обеспечивает асинхронный вызов функций обработки потоков данных, привязанных к каналам связи, на узлах с установленными графическими ускорителями (видеокартами). На каждом запуске такой функции модулю сообщаются уникальные идентификаторы карт, которые могут быть использованы для организации вычислений. Ускорители в рамках системы считаются неразделяемым ресурсом, диспетчер нагрузки планирует исполнение задач так, чтобы обеспечить монопольный доступ. Однако, идентификаторы представляют собой лишь строки, программист, реализующий прикладной модуль, сам решает, как эти строки интерпретировать, какую технологию использовать (CUDA, OpenCL или какую-то другую). При желании можно запустить множество независимых потоков обработки ресурсов, поступающих по каналам передачи данных, используя для этого систему всего лишь с одним ускорителем вычислений.

Вычислитель (Worker) — программный компонент системы, обеспечивающий дополнительный уровень абстракции над аппаратным уровнем. Представляет собой реализацию пула потоков, способного загружать и выполнять код, реализованный в программных модулях, в соответствии с указаниями диспетчера распределения нагрузки. Вычислители, запущенные на доступном оборудовании, образуют вычислительную сеть. Управление ходом вычислений осуществляется диспетчером распределения нагрузки, данную роль может принять любой из активных вычислителей. Распределение роли управляющего узла повышает отказоустойчивость системы. В ходе инициализации вычислитель сообщает системе диспетчеризации о доступных аппаратных ресурсах (количество вычислительных ядер, идентификаторы GPU). Эта информация используется диспетчером распределения нагрузки при назначении задач.

Для обеспечения отказоустойчивости вычислители постоянно (по таймеру) сообщают диспетчеру о своем состоянии, в случае, если подобные сообщения не поступают длительное время, диспетчер

принимает решение о перераспределении задач. Вычислители поддерживают локальный LRU-кэш для хранения запрошенных из БД ресурсов, что позволяет уменьшить количество запросов к БД и объем передаваемой по сети информации. Особым типом вычислителя можно считать вычислитель, запущенный на стороне клиента, ресурсы подобных вычислителей не распределяются автоматически и могут быть использованы лишь для запуска определенного пользователем списка модулей, в том числе модулей визуализации.

Диспетчер распределения нагрузки (Scheduler) предназначен для генерации управляющих команд и распределения задач между вычислителями. Новые задачи и управляющие команды порождаются в ходе обработки прикладной задачи по факту возникновения в выходных каналах модулей новых ресурсов или возникновения особого состояния системы (например, восстановления работоспособности после сбоя). Диспетчер обеспечивает функционирование схемы решения задачи, определяя, на каких вычислителях будет произведена обработка данных, в соответствии с заданными ограничениями, в том числе используя информацию об ограничениях на распределение модулей по вычислительным узлам; метаинформацию модуля, описывающую принцип его функционирования; информацию о доступных аппаратных ресурсах; приоритеты модулей. Запуск каждого модуля, входящего в схему решения задачи считается атомарной операцией (транзакцией) и в случае отказа узла происходит отмена всех назначенных ранее вычислителю задач и их перераспределение.

Файл описания прикладной задачи хранит информацию о требуемых для решения задачи модулях, их настройках и определенных пользователем каналах передачи данных. Структурно описание задачи состоит из двух основных секций: секции описания модулей (определены тегом <modules>), которые необходимо загрузить вычислительному ядру, с указанием параметров их инициализации; секции описания каналов передачи данных (определена тегом <channels>). Дополнительно могут быть указаны ограничения на расположение модулей, любой из модулей может быть «привязан» к конкретному вычислителю или группе вычислителей. Частным случаем подобных ограничений является возможность привязки модулей чтения данных и сохранения результатов к вычислителю, запущенному на стороне клиента, что позволяет организовать схему обработки, в которой данные поступают

с клиентской машины, обрабатываются в распределенной вычислительной среде, после чего результаты обработки поступают клиенту для отображения в графическом интерфейсе и/или для сохранения на диск.

Состояние системы, включая описание пользовательских задач, текущее состояние системы диспетчеризации, обрабатываемые в ходе решения задач данные и информационные сообщения, сохраняется в распределенной СУБД в виде множества таблиц. Все информационные сообщения о функционировании системы и ходе решения задач пользователей сохраняются в БД, все сообщения снабжены временными метками и имеют привязку к уникальному идентификатору задачи, что позволяет системе функционировать в многопользовательском режиме.

Пользователь сам строит схему задачи, используя принципы визуально-блочного проектирования. В состав разработанной системы входят

- вычислительное ядро для работы в гетерогенной вычислительной среде [**24**];
- программные модули предварительной и нейросетевой обработки данных;
- универсальный графический интерфейс [**25**], дополненный функциональной поддержкой отказоустойчивого вычислительного ядра (см. рис. 2, 3 и 4).

Обобщая, перечислим основные особенности системы, так или иначе связанные с реализованной схемой обеспечения отказоустойчивости:

- для того, чтобы вычислительное ядро могло отличать одну сессию от другой, при инициализации задания передается уникальный идентификатор сессии, позволяющий не только снимать строго определенную задачу (прекращать счет), но и подключаться к ранее запущенным задачам;
- функции визуализации, в том числе когнитивной, исполняются непосредственно вычислительными модулями, а не основным графическим интерфейсом;
- вычислительное ядро на уровне каналов передачи данных работает с нетипизированными данными; ядро не формирует программных низкоуровневых каналов передачи данных, модули получают лишь те поля «передаваемых» ресурсов, что им необходимы, строго через прикладной интерфейс базы данных Apache Cassandra, то есть имеется поддержка стримминга данных.

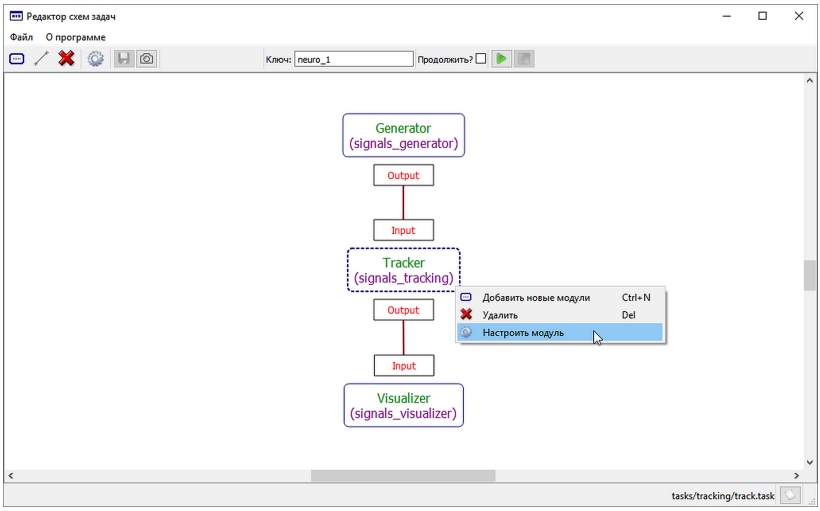

Рис. 2. Главное окно графического интерфейса системы

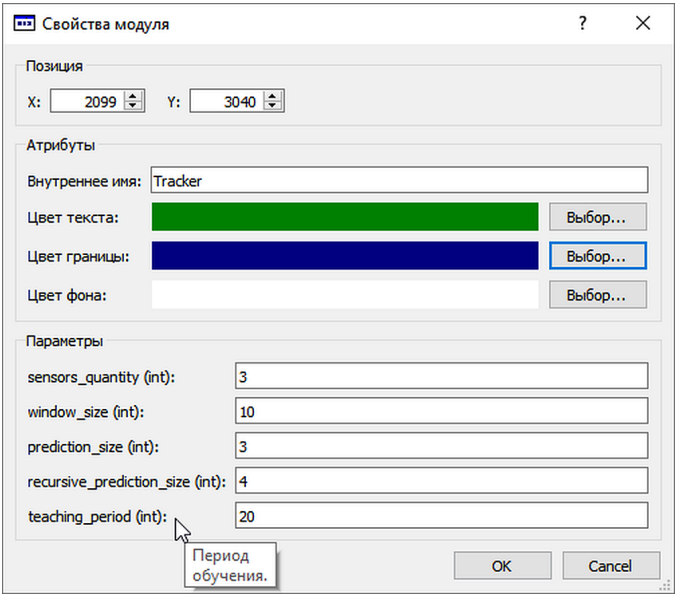

Рис. 3. Форма настроек модуля

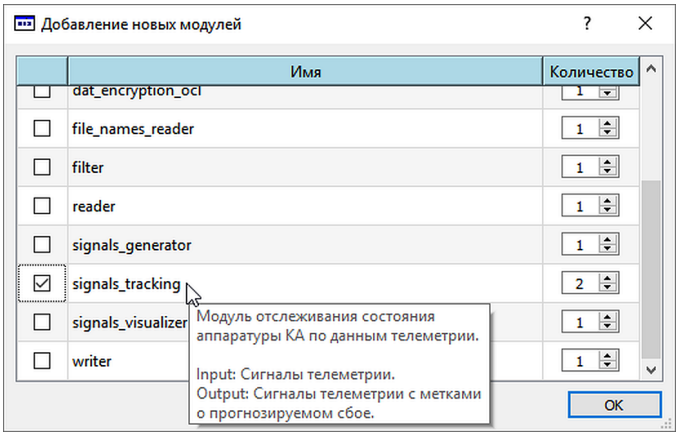

Рис. 4. Форма добавления новых модулей в схему

2. Когнитивная визуализация в составе отказоустойчивой системы

Актуальность технологии когнитивной визуализации обуславливается необходимостью оперативной оценки контролируемых ситуаций по данным для поддержки принятия решений человека-оператора. Перспективным направлением исследований служит когнитивная визуализация многомерных данных специальными средствами интерфейса, позволяющая оперативно обнаруживать нештатные ситуации. Человеко-машинное взаимодействие строится на базе технологий образного представления больших объемов информации контролируемых объектов, способствующих быстрому принятию решений. Результатом внедрения когнитивной графики является повышение эффективности информационной поддержки лица, принимающего решения.

Использование методов обеспечения отказоустойчивости положительным образом сказывается на поддержке лиц, принимающих решения, так как система обеспечивает бесшовный режим работы пользователя. Например, модуль визуализации может интерактивно подгружать данные от модулей, осуществляющих обработку целевых данных. В случае выхода из строя тех или иных вычислительных узлов, выполняющих обработку данных, процесс визуализации в рамках отказоустойчивой системы не прерывается. Таким образом, когнитивную визуализацию можно считать индикатором работоспособности

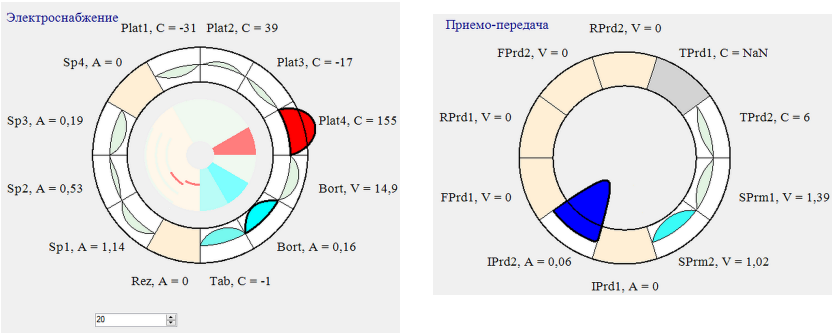

Рис. 5. Мониторинг состояния подсистем космического аппарата

рассматриваемой системы.

На рис. 5 приведены примеры когнитивных образов, получаемых от соответствующих модулей визуализации в составе разработанной системы, отображающих состояния подсистем исследованного в рамках работы космического аппарата [26–28]. На рис. 6 приведен скриншот модуля визуализации ишемизированного мозга лабораторного животного (крысы), где программно выделенные фрагменты ишемического поражения мозга представлены отдельными 3D-объектами [29, 30]. На рис. 7 — результат работы нейросетевого модуля поиска целевых объектов на панорамном изображении дистанционного зондирования Земли [31].

Несмотря на то, что приведенные когнитивные образы затрагивают разные прикладные области, все они отражают процессы обработки информации реального времени в динамике и служат индикатором работоспособности вычислительной системы при своем непрерывном воспроизведении.

Заключение

Новая программная система имеет три основные составляющие: универсальное отказоустойчивое вычислительное ядро; графический интерфейс с функциями создания схем вычислительных задач из ранее разработанных библиотечных модулей; подсистема визуализации, позволяющая исполнять процессы визуализации, в том числе когнитивной, на заранее определенных узлах. Программная система

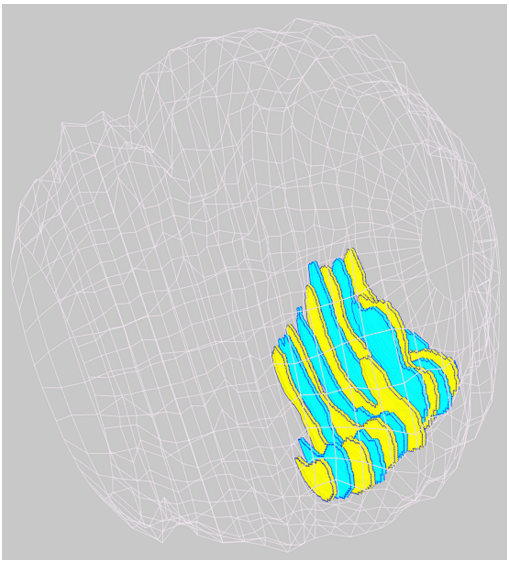

Рис. 6. 3D-модель ишемизированного мозга крысы

Рис. 7. Визуализация результатов поиска целевой техники военного назначения

апробирована на широком ряде вычислительных задач и показала свою эффективность и удобство использования.

Проведенные испытания показали полную защищенность от сбоев. Вычисления идут вплоть до полного отключения всех компьютеров вычислительной сети, решение поставленных задач продолжается при восстановлении доступа к базе данных.

Список литературы

[1] В. А. Васенин, В. А. Роганов. «GRACE: распределенные приложения в Internet», *Открытые системы*, 2001, №5, с. 29–33. ↑86

[2] В. Ю. Гришин, А. В. Лобанов, В. Г. Сиренко. «Сетецентризм и отказоустойчивость», *Труды XII Всероссийского совещания по проблемам управления (ВСПУ-2014)* (Москва, 16-19 июня 2014 г.), с. 6855–6864, ✻. ↑86

[3] А. А. Кузнецов, В. А. Роганов. «Поддержка топологии вычислительного пространства в системе OpenTS», *Программные системы: теория и приложения*, 2010, №3, с. 93–106, ✻. ↑86

[4] В. А. Богатырев. «Отказоустойчивость распределенных вычислительных систем динамического распределения запросов и размещение функциональных ресурсов», *Наука и образование: научное издание МГТУ им. Н.Э. Баумана*, 2006, №1, URL ✻. ↑87

[5] В. П. Корячко, С. В. Скворцов, В. И. Шувиков. «Характеризация диагностических графов для симметричной модели дешифрации синдрома», *Наука и образование: научное издание МГТУ им. Н.Э. Баумана*, 2006, №4, URL ✻. ↑87

[6] В. А. Мелентьев. «Функция структурной отказоустойчивости и d-ограниченная компонента связности графа вычислительной системы», *Прикладная дискретная математика*, 2008, №2, с. 102–106, ✻. ↑87

[7] Е. Н. Новиков, Р. Р. Валеев. «Резервирование сервера технологических данных — путь к созданию отказоустойчивых технологических серверов», *Промышленные АСУ и контроллеры*, 2002, №7, с. 14–18. ↑87

[8] А. К. Ландман, А. Э. Петров, О. О. Сакаев, М. В. Петрушков, А. В. Субботин-Чукальский. «Подходы к разработке отказоустойчивой распределенной вычислительной системы противоаварийного управления», *Научные проблемы транспорта Сибири и Дальнего Востока*, 2014, №4, с. 335–340, ✻. ↑88

[9] А. А. Бондаренко, М. В. Якобовский. «Обеспечение отказоустойчивости высокопроизводительных вычислений с помощью локальных контрольных точек», *Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика*, 2014, №3, с. 20–36, ✻. ↑88

[10] Е. О. Тютляева, А. А. Московский. «Методы обеспечения отказоустойчивости в библиотеке шаблонных классов C++ для распараллеливания T-Sim», *Программные системы: теория и приложения*, 2011, №3, с. 17–28, . ↑89

[11] А. В. Штенцель. *Инструментальные средства формирования мультиверсионной архитектуры отказоустойчивых программных систем*, Диссертация на соискание ученой степени кандидата технических наук, Красноярск, 2008, 183 с., ↑90

[12] Ю. В. Борисенко. *Метод, алгоритмы и аппаратные средства оперативного переразмещения программ в отказоустойчивых мультикомпьютерных системах*, Диссертация на соискание ученой степени кандидата технических наук, Курск, 2014, 128 с., ↑90

[13] А. Н. Фирсов. «Оптимизация на основе статистических данных асинхронной распределенной системы, устойчивой к произвольным отказам», Международная научная конференция «Параллельные вычислительные технологии (ПаВТ) 2009», 2009, URL ↑90

[14] А. Н. Фирсов. «Архитектура распределенной системы устойчивой к произвольным отказам», Девятая международная конференция-семинар «Высокопроизводительные параллельные вычисления на кластерных системах», 2009, URL ↑91

[15] M. Hovestadt. “Fault tolerance mechanisms for sla-aware resource management”, *Parallel and Distributed Systems. Proceedings. 11th International Conference on. vol. 2* (Fukuoka, Japan, July 20-22, 2005), doi ↑91

[16] F. Heine, M. Hovestadt, O. Kao, A. Keller. “SLA-aware Job Migration in Grid Environments”, *Grid Computing: New Frontiers of High Performance Computing*, Elsevier, pp. 185–201, doi ↑91

[17] M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly. “Dryad: Distributed Data-parallel Programs from Sequential Building Blocks”, *Proceedings of the 2007 Eurosys Conference* (Lisboa, Portugal, March 21-23, 2007), doi ↑91

[18] Y. Yu, M. Isard, D. Fetterly, M. Budiu, Ú. Erlingsson, P. K. Gunda, J. Currey. “DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language”, *OSDI’08: Eighth Symposium on Operating System Design and Implementation USENIX* (San Diego, California, USA, December 8-19, 2007), URL ↑91

[19] J. Ekanayake, T. Gunarathne, G. Fox, A. S. Balkir, C. Poulain, N. Araujo, R. Barga. “DryadLINQ for Scientific Analyses”, *e-Science ’09. Fifth IEEE International Conference on* (Oxford, UK, December 9-11, 2009), doi ↑91

[20] F. McSherry, T. Rodeheffer. *Using DryadLINQ for Large Matrix Operations*, Microsoft Technical Report, 2011, URL ↑91

[21] D. C. Aiftimiei, M. Antonacci, S. Bagnasco, T. Boccali, R. Bucchi, M. Caballer, A. Costantini, G. Donvito, L. Gaido, A. Italiano. “Geographically distributed

Batch System as a Service: the INDIGO-DataCloud approach exploiting HTCondor", *Journal of Physics: Conference Series*, 2017, ↑91

[22] G. Suo, Y. Lu, X. Liao, M. Xie, H. Cao. "NR-MPI: A non-stop and fault resilient mpi supporting programmer defined data backup and restore for e-scale super computing systems", *Supercomputing frontiers and innovations*, 2016, no.1, pp. 4–21, ↑92

[23] S. Kalid, A. Syed, A. Mohammad, M. N. Halgamuge. "Big-Data NoSQL Databases: A Comparison and Analysis of "Big-Table", "DynamoDB", and "Cassandra"", *Proceedings of the IEEE 2nd International Conference on Big Data Analysis (ICBDA'17)* (Beijing, China, March 10-12, 2017), pp. 89–93, ↑92

[24] А. А. Талалаев, В. П. Фраленко. «Архитектура комплекса конвейерно-параллельной обработки данных в гетерогенной вычислительной среде», *Вестник Российского университета дружбы народов. Серия Математика. Информатика. Физика*, 2013, №3, с. 113–117, ↑97

[25] В. П. Фраленко. «Универсальный графический интерфейс визуального проектирования параллельных и параллельно-конвейерных приложений», *Программные системы: теория и приложения*, 2016, №3, с. 45–70, ↑97

[26] Н. С. Абрамов, А. А. Талалаев, В. П. Фраленко. «Интеллектуальный анализ телеметрической информации для диагностики оборудования космического аппарата», *Информационные технологии и вычислительные системы*, 2016, №1, с. 64–75, ↑100

[27] Н. С. Абрамов, А. А. Талалаев, В. П. Фраленко, В. М. Хачумов, О. Г. Шишкин. «Высокопроизводительная нейросетевая система мониторинга состояния и поведения подсистем космических аппаратов по телеметрическим данным», *Программные системы: теория и приложения*, 2017, №3, с. 109–131, ↑100

[28] Ю. Г. Емельянова. «Разработка методов когнитивного отображения состояний динамических систем реального времени», *Искусственный интеллект и принятие решений*, 2016, №3, с. 21–30, ↑100

[29] В. П. Фраленко, М. В. Хачумов, М. В. Шустова. «Выделение и когнитивная визуализация трансплантированных мезенхимальных стволовых клеток на снимках магнитно-резонансной томографии», *Искусственный интеллект и принятие решений*, 2017, №3, с. 10–20, ↑100

[30] В. П. Фраленко, М. В. Шустова. «Программный комплекс для автоматического выделения, визуализации и расчета информативных характеристик областей интереса в биомедицинских данных МРТ», *Вестник новых медицинских технологий, электронный журнал*, 2017, №4, ↑100

[31] В. П. Фраленко. «Экспериментальное исследование возможностей нейронной сети типа «Darknet» на задаче обработки снимков дистанционного зондирования», *Авиакосмическое приборостроение*, 2017, №6, с. 44–52, ↑100

Рекомендовал к публикации *д.т.н. В. М. Хачумов*

Пример ссылки на эту публикацию:

А. А. Талалаев, В. П. Фраленко. «Отказоустойчивая система организации высокопроизводительных вычислений для решения задач обработки потоков данных». *Программные системы: теория и приложения*, 2018, **9**:1(36), с. 85–108. 10.25209/2079-3316-2018-9-1-85-108
http://psta.psiras.ru//read/psta2018_1_85-108.pdf

Об авторах:

Александр Анатольевич Талалаев

К.т.н., старший научный сотрудник ИЦМС ИПС им. А. К. Айламазяна РАН, автор более 40 публикаций. Область научных интересов: искусственный интеллект, машинная графика, распознавание образов, параллельные вычисления.

0000-0003-0186-9499
e-mail: arts@arts.botik.ru

Виталий Петрович Фраленко

К.т.н., ведущий научный сотрудник ИЦМС ИПС им. А.К. Айламазяна РАН. Область научных интересов: интеллектуальный анализ данных и распознавание образов, искусственный интеллект и принятие решений, параллельные алгоритмы, сетевая безопасность, диагностика сложных технических систем, графические интерфейсы, блокчейн-технологии.

0000-0003-0123-3773
e-mail: alarmod@pereslavl.ru

UDC 004.75:004.052.3:004.042

Alexander Talalaev, Vitaly Fralenko. *Fault-tolerant system for organizing high-performance computing for solving data processing problems.*

Abstract. The paper presents an overview of existing systems of organization of fault-tolerant computing; functional characteristics of the developed high-performance system based on the computational core, a special interface and application modules are considered. At the heart of the proposed system of computing is distributed NoSQL database Apache Cassandra, providing mechanisms for fault-tolerant storage and automatic replication of data in a heterogeneous computing environment. The system, equipped with a special graphical interface, allows to develop solutions for various application areas. Plugins can also perform the functions of visualizing data streams (*In Russian*).

Key words and phrases: computing organization, fault tolerance, database, graphical interface, module, visualization, heterogeneous environment.

References

[1] V. A. Vasenin, V. A. Roganov. "GRACE: Distributed Applications on the Internet", *Open Systems*, 2001, no.5, pp. 29–33 (in Russian)

[2] V. Yu. Grishin, A. V. Lobanov, V. G. Sirenko. "Network Centrism and Fault tolerance", *Trudy XII Vserossijskogo soveshhanija po problemam upravlenija (VSPU-2014)* (Moscow, June 16-19, 2014), pp. 6855–6864 (in Russian)

[3] A. A. Kuznetsov, V. A. Roganov. "Network Topology Support in the OpenTS Programming System", *Program Systems: Theory and Aplications*, 2010, no.3, pp. 93–106 (in Russian)

[4] V. A. Bogatyrev. "Fault Tolerance of Distributed Computing Systems for Dynamic Query Distribution and Allocation of Functional Resources", *Science & Education: scientific edition of Bauman MSTU*, 2006, no.1 (in Russian),

[5] V. P. Korjachko, S. V. Skvorcov, V. I. Shuvikov. "Characterization of Diagnostic Graphs for a Symmetric Model of Syndrome Decoding", *Science & Education: scientific edition of Bauman MSTU*, 2006, no.4 (in Russian),

[6] V. A. Melentiev. "The Function of Structural Fault Tolerance and d-bounded Component of the Computer System Connectivity Graph", *Applied Discrete Mathematics*, 2008, no.2, pp. 102–106 (in Russian)

[7] E. N. Novikov, R. R. Valeev. "Reservation of the Technological Data Server — the Way to Create Fault-tolerant Technology Servers", *Industrial Automatic Control Systems and Controllers*, 2002, no.7, pp. 14–18 (in Russian)

10.25209/2079-3316-2018-9-1-85-108

[8] A. K. Landman, A. Je. Petrov, O. O. Sakaev, M. V. Petrushkov, A. V. Subbotin-Chukal'skij. "Approaches to the Development of a Fault-tolerant Distributed Computer Emergency Control System", *Scientific Transport Problems in Siberia and Far East*, 2014, no.4, pp. 335–340 (in Russian)

[9] A. A. Bondarenko, M. V. Iakobovski. "Fault Tolerance for HPC by Using Local Checkpoints", *Bulletin of the South Ural State University. Series "Computational Mathematics and Software Engineering"*, 2014, no.3, pp. 20–36 (in Russian)

[10] E. O. Tjutljaeva, A. A. Moskovskij. "T-Sim Fault Tolerance", *Program Systems: Theory and Aplications*, 2011, no.3, pp. 17–28 (in Russian)

[11] A. V. Shtencel. *Tools for the Formation of a Multiversion Architecture of Fault-tolerant Software Systems*, Dissertacija na soiskanie uchenoj stepeni kandidata tehnicheskih nauk, Krasnoyarsk, 2008 (in Russian), 183 p.

[12] Ju. V. Borisenko. *Method, Algorithms and Hardware for the Rapid Reallocation of Programs in Fault-tolerant Multicomputer Systems*, Dissertacija na soiskanie uchenoj stepeni kandidata tehnicheskih nauk, Kursk, 2014 (in Russian), 128 p.

[13] A. N. Firsov. "Optimization Based on Statistical Data of an Asynchronous Distributed System, Resistant to Arbitrary Failures", International Scientific Conference "Parallel Computing Technologies (PAVT) 2009", 2009 (in Russian), URL

[14] A. N. Firsov. "Resistant to Arbitrary Failures Distributed System Architecture", Ninth International Conference -Seminar "High Performance Parallel Computing on Cluster Systems, 2009 (in Russian), URL

[15] M. Hovestadt. "Fault tolerance mechanisms for sla-aware resource management", *Parallel and Distributed Systems. Proceedings. 11th International Conference on. vol. 2* (Fukuoka, Japan, July 20-22, 2005)

[16] F. Heine, M. Hovestadt, O. Kao, A. Keller. "SLA-aware Job Migration in Grid Environments", *Grid Computing: New Frontiers of High Performance Computing*, Elsevier, pp. 185–201

[17] M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly. "Dryad: Distributed Data-parallel Programs from Sequential Building Blocks", *Proceedings of the 2007 Eurosys Conference* (Lisboa, Portugal, March 21-23, 2007)

[18] Y. Yu, M. Isard, D. Fetterly, M. Budiu, Ú. Erlingsson, P. K. Gunda, J. Currey. "DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language", *OSDI'08: Eighth Symposium on Operating System Design and Implementation USENIX* (San Diego, California, USA, December 8-19, 2007), URL

[19] J.~Ekanayake, T.~Gunarathne, G.~Fox, A. S.~Balkir, C.~Poulain, N.~Araujo, R.~Barga. "DryadLINQ for Scientific Analyses", *e-Science '09. Fifth IEEE International Conference on* (Oxford, UK, December 9-11, 2009), DOI

[20] F.~McSherry, T.~Rodeheffer. *Using DryadLINQ for Large Matrix Operations*, 2011, URL

[21] D. C.~Aiftimiei, M.~Antonacci, S.~Bagnasco, T.~Boccali, R.~Bucchi, M.~Caballer, A.~Costantini, G.~Donvito, L.~Gaido, A.~Italiano. "Geographically distributed Batch System as a Service: the INDIGO-DataCloud approach exploiting HTCondor", *Journal of Physics: Conference Series*, 2017, URL

[22] G.˜Suo, Y.˜Lu, X.˜Liao, M.˜Xie, H.˜Cao. “NR-MPI: A non-stop and fault resilient mpi supporting programmer defined data backup and restore for e-scale super computing systems”, *Supercomputing frontiers and innovations*, 2016, no.1, pp. 4–21,

[23] S.˜Kalid, A.˜Syed, A.˜Mohammad, M. N.˜Halgamuge. “Big-Data NoSQL Databases: A Comparison and Analysis of “Big-Table”, “DynamoDB”, and “Cassandra””, *Proceedings of the IEEE 2nd International Conference on Big Data Analysis (ICBDA'17)* (Beijing, China, March 10-12, 2017), pp. 89–93,

[24] A. A.˜Talalaev, V. P.˜Fralenko. “The Architecture of a Parallel-pipeline Data Processing Complex for Heterogeneous Computing Environment”, *Bulletin of the Russian University of Peoples' Friendship. Series Mathematics. Computer science. Physics*, 2013, no.3, pp. 113–117 (in˜Russian)

[25] V. P. Fralenko. “Universal Graphical User Interface for Visual Design of Parallel and Parallel-pipelined Applications”, *Program Systems: Theory and Aplications*, 2016, no.3, pp. 45–70 (in Russian),

[26] N. S.˜Abramov, A. A.˜Talalaev, V. P.˜Fralenko. “Intelligent Telemetry Data Analysis for Diagnosing of the Spacecraft Hardware”, *Journal of Information Technologies and Computing Systems*, 2016, no.1, pp. 64–75 (in˜Russian)

[27] N. S. Abramov, A. A. Talalaev, V. P. Fralenko, V. M. Hachumov, O. G. Shishkin. “The High-performance Neural Network System for Monitoring of State and Behavior of Spacecraft Subsystems by Telemetry Data”, *Program Systems: Theory and Aplications*, 2017, no.3, pp. 109–131 (in Russian),

[28] Ju. G.˜Emeljanova. “Development of Cognitive Representation Methods for Real Time Dynamic Systems”, *Artificial Intelligence and Decision Making*, 2016, no.3, pp. 21–30 (in˜Russian)

[29] V. P. Fralenko, M. V. Khachumov, M. V. Shustova. “Allocation and Cognitive Visualization of Transplanted Mesenchymal Stem Cells in Images of a Magnetic Resonance Tomography”, *Artificial Intelligence and Decision Making*, 2017, no.3, pp. 10–20 (in Russian)

[30] V. P. Fralenko, M. V. Shustova. “Program Complex for Automatic Localization, Visualization and Calculation of Informative Characteristics of Interest Areas in Biomedical Data of MRI”, *Journal of New Medical Technologies, eEdition*, 2017, no.4 (in Russian),

[31] V. P. Fralenko. “Experimental Investigation with “Darknet” Neural Network Capabilities on the Task of Remote Sensing Images Processing”, *Aerospace Instrument-Making*, 2017, no.6, pp. 44–52 (in Russian)

Sample citation of this publication:

Alexander Talalaev, Vitaly Fralenko. “Fault-tolerant system for organizing high-performance computing for solving data processing problems”. *Program Systems: Theory and Applications*, 2018, **9**:1(36), pp. 85–108. (*In Russian*).

10.25209/2079-3316-2018-9-1-85-108

`http://psta.psiras.ru//read/psta2018_1_85-108.pdf`