УДК 004.75
DOI 10.21685/2072-3059-2018-2-4

А. Н. Токарев

УМЕНЬШЕНИЕ НАГРУЗКИ И ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ВЫЧИСЛЕНИЙ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ С НЕНАДЕЖНЫМИ УЗЛАМИ

Аннотация.

Актуальность и цели. Объектом исследования являются распределенные вычислительные системы кластерного и кластерно-метакомпьютерного типа без постоянной связи между узлами. Предметом исследования является вычислительный процесс в распределенных системах с ненадежными узлами. Цель работы – уменьшить влияние административных отказов на протекание процесса вычислений в распределенной системе, а также уменьшить нагрузку на сеть и на вычислительные узлы и, таким образом, увеличить производительность.

Материалы и методы. Исследование и разработка стратегии повышения производительности в распределенных системах с ненадежными узлами выполнены с использованием теории надежности, теории марковских случайных процессов.

Результаты. Произведен анализ надежностных факторов, влияющих на производительность систем распределенных вычислений. Предложена классификация отказов в зависимости от аппаратных или административных причин. Рассмотрены имеющиеся системы, предотвращающие ошибки вычислений вследствие отказов разного рода. Предложена стратегия повышения производительности в системах распределенных вычислений с ненадежными узлами, позволяющая уменьшить влияние административных отказов на протекание процесса вычислений в распределенной системе, а также уменьшить нагрузку на сеть и увеличить производительность системы в целом.

Выводы. Анализ имеющихся систем предотвращения вычислительных ошибок и поддержания корректности вычислительного процесса в распределенных вычислительных системах позволил сделать выводы о том, что имеется возможность без использования специальных аппаратных средств обеспечить достаточную производительность и отказоустойчивость системы с ненадежными узлами. Для этого предложена соответствующая стратегия.

Ключевые слова: распределенная система, вычисления, кластер, подзадача, надежность, отказоустойчивость.

A. N. Tokarev

LOAD REDUCING AND CAPABILITY RISE OF CALCULATIONS IN DISTRIBUTED COMPUTATIONAL SYSTEMS WITH UNRELIABLE NODES

Abstract.

Background. The research deals with the study of distributed computing systems of cluster and cluster-metacomputer type without a constant link between nodes. The subject of research is the computational process in distributed systems with unreliable nodes. The purpose of the study is to reduce the impact of administrative failures on the progress of the computation process in a distributed system, and to reduce the load on the network and on the compute nodes and, thus, increase productivity.

Materials and methods. Research and development of a strategy for increasing performance in distributed systems with unreliable nodes is performed using the theory of reliability, the theory of Markov random processes.

Results. The analysis of reliability factors affecting the performance of distributed computing systems is made. The classification of failures is proposed depending on hardware or administrative reasons. Existing systems that prevent errors in calculations due to failures of various kinds are considered. A strategy for improving performance in distributed computing systems with unreliable nodes is proposed, which makes it possible to reduce the effect of administrative failures on the process of computing in a distributed system, and reduce the load on the network and increase the overall system performance.

Conclusions. Comparison of descriptive capabilities of the two computer models of the collision process of the dummy with the windshield of the car proved that the second computer model satisfies the requirements of the established quality representations of the process of deformation and fracture of laminated safety glass in modern passenger cars. Analysis of existing systems preventing computational errors and maintaining the correctness of the computational process in distributed computing systems led to the conclusion that it is possible to provide sufficient performance and fault tolerance of a system with unreliable nodes without the use of special hardware. For this, an appropriate strategy has been proposed.

Key words: distributed system, calculations, cluster, subtask, reliability, fault tolerance.

При планировании решения задач в распределенных вычислительных системах метакомпьютерного и кластерно-метакомпьютерного типа разработчик сталкивается с проблемой надежного получения результата. Особенно это актуально в децентрализованных гетерогенных системах, имеющих в своем составе разнородные вычислительные узлы. Примером таких систем являются грид-системы, основанные на принципах Desktop Grid и состоящие в основной массе из обычных персональных компьютеров, которые сами пользователи предоставляют на добровольной основе для той или иной распределенной задачи.

Естественно, в таких системах не может быть речи о каком-то централизованном контроле за аппаратными средствами и запущенном на узлах программном обеспечении. Персональный компьютер (ПК), предоставленный пользователем для проекта, может иметь аппаратные проблемы (сбойную память, дисковые устройства с bad-блоками, сбойные чипсеты и контроллеры устройств ввода-вывода и т.п.). Ни одно устройство не застраховано от ошибок при расчете, передаче и хранении информации. Все это приводит к тому, что подзадача, выданная данному узлу, может быть решена некорректно. Таким образом, первый фактор, который следует учитывать, – *аппаратный*.

Кроме того, в рассматриваемых грид-системах отсутствует возможность централизованно управлять доступностью узлов и временем их непрерывной работы. Это значит, что выданная узлу подзадача с некоторой ненулевой (и иногда достаточно высокой) вероятностью не будет решена вовремя из-за того, что узел может быть перезагружен, выключен пользователем, на узле запущена другая задача, занимающая полностью процессорное время, и т.д. Особенно критичным этот фактор является в сильно связных кластерно-метакомпьютерных системах, где решение задачи – гораздо менее длительный процесс, чем в классических Desktop Grid, и где каждый узел вносит свой весомый вклад. Это позволяет говорить о втором факторе обеспечения надежности, напрямую не связанном с техникой и аппаратным обеспечением. Назовем его *административным*.

Для учета аппаратного фактора можно использовать как аппаратные, так и программные методики обеспечения надежности вычислений и валидации результата.

Аппаратные средства контроля

К аппаратным средствам контроля можно условно отнести также аппаратно-программные средства и программные средства низкого уровня, к которым разработчик распределенных приложений не имеет непосредственного доступа.

В последние несколько лет крупными компаниями-производителями аппаратных средств ПК и контроллеров уделяется повышенное внимание вопросу контроля целостности данных и защите вычислительной системы от сбоев и непредумышленных либо предумышленных ошибок. Корпорация Intel предложила для этих целей свою аппаратно-программную систему, которая называется Intel Trusted Execution Technology (TXT). В свое время корпорации AMD и Intel уже осуществили успешное внедрение в массовые ПК технологий, защищающих программный код от ошибок или действий вредоносного кода. Это так называемый NX Bit (в терминологии Intel – XD Bit). Это бит запрета исполнения, который является атрибутом страницы памяти в таблице страниц. Бит определяет, что определенная страница является хранилищем данных, а не кода. В этом случае несанкционированное исполнение кода в этой странице запрещено. Ранее уязвимость подобного рода часто эксплуатировалась вредоносными программами при атаке так называемым способом переполнения буфера или некоторыми другими. Также в результате программных ошибок или сбоев времени выполнения программы может быть инициирован переход в другие области памяти. В случае использования бита контроля код за пределами разрешенной области сегмента кода уже не может быть исполнен, что дает некоторую степень контроля за процессом выполнения программы. Когда возникает подобная ситуация, управление посредством прерывания с сигналом SIGSEGV (выход за пределы сегмента) передается операционной системе либо управляющей программе, которая сможет проанализировать произошедшую ситуацию и соответствующим образом на нее отреагировать.

Технология защиты Intel TXT идет еще дальше. Ее работа основана на использовании аппаратного контроллера Trusted Platform Module, который осуществляет безопасную загрузку операционной системы в изолированной

области и выполнение программного кода в изолированной среде. Кроме того, возможно использование отказоустойчивого шифрования данных на всех этапах их ввода-вывода и хранения. Технология Intel TXT имеет встроенную систему самоконтроля, так называемую аттестацию, проходя которую, все компоненты TXT проверяются на целостность.

Всего в технологии несколько компонентов, с точки зрения отказоустойчивого исполнения кода нас интересуют прежде всего два:

1) защищенный запуск (Protected Launch): эта функция контролирует и защищает критические части операционной системы и другие, связанные с системой компоненты от несанкционированного доступа во время запуска. Например, компоненты ядра системы защищаются во время и после запуска;

2) защищенное выполнение (Protected Execution): эта возможность разрешает приложениям выполняться в изолированных окружениях, чтобы оставаться максимально защищенными от другого программного обеспечения, работающего на этой же платформе. Каждое приложение, которое выполняется в этом режиме, имеет доступ к своим собственным физически разделенным ресурсам.

Таким образом, аппаратно-программная технология Intel TXT, прежде всего направленная на контроль за целостностью кода и являющаяся неким «аппаратным антивирусом», помогает за счет отказоустойчивого шифрования и контроля за исполнением кода решить и смежную задачу – поддержать отказоустойчивый процесс вычислений.

Что касается чисто аппаратного контроля, то вычислительные узлы в этом случае должны содержать ряд соответствующих схематических и конструктивных решений:

– источники бесперебойного питания с аккумуляторной батареей;

– дублированные блоки питания (минимум 2);

– оперативная память с контролем четности (ECC);

– RAID-контроллеры дисков уровней RAID1 (зеркалирование) или RAID10 (зеркалирование с чередованием);

– дисковые контроллеры с резервной батареей;

– сетевые контроллеры с модулями проверки целостности данных и т.д.

Однако, поскольку нами рассматриваются распределенные вычислительные сети, наложенные на имеющуюся инфраструктуру вычислительной техники, о каком-либо достаточном распространении таких аппаратных средств говорить пока не приходится, так как они представлены в среде персональных компьютеров достаточно слабо.

Программные средства контроля

Чисто программные средства контроля вычислительного процесса в распределенных вычислительных системах можно разделить на две основные категории:

1) средства восстановления после отказа;

2) средства предотвращения отказа.

Средства восстановления после отказа могут либо осуществлять ограниченную коррекцию результата после обнаружения ошибки (если это возможно), либо полностью откатывать транзакцию по получению результата некорректно решенной подзадачи. Вследствие частности и ограниченности первого подхода, а также широкого диапазона решаемых задач, не позволя-

ющего применить универсальные средства коррекции, в большинстве распределенных вычислительных систем чаще всего используется второй подход.

Недостаток данного способа – вследствие необходимости повторного расчета потерянной подзадачи потери производительности могут быть чрезмерно велики, особенно в рассматриваемых системах с низкой административной надежностью.

Средства предотвращения отказа, т.е. применительно к распределенным вычислительным системам, – это прежде всего дублирование вычислительных узлов и *репликация подзадач*.

При этом подзадача, выданная одному вычислительному узлу, выдается другому вычислительному узлу (или нескольким узлам), и после окончания расчета сервер сравнивает результаты. Если они совпадают либо отличаются на величину допустимого отклонения, то результат принимается верным и подзадача считается решенной. Недостаток такого подхода очевиден – избыточность расчетов, необходимость загрузки узлов одинаковыми подзадачами, что приводит к падению производительности системы до уровня 50 % от максимально достижимой, а в некоторых случаях и менее.

Для уменьшения избыточности можно воспользоваться методом адаптивной репликации. Например, в вычислительных сетях BOINC [1] данный метод заключается в следующем. Для каждого вычислительного узла h сервер хранит его коэффициент ненадежности $e(h)$. Изначально $e(h) = 0{,}1$. В дальнейшем его значения рассчитываются следующим образом:

$e(h) = e(h) \cdot 0{,}95$ – если результат решения подзадачи был признан верным;

$e(h) = e(h) + 0{,}1$ – если результат решения подзадачи был признан ошибочным.

Таким образом, коэффициент $e(h)$ представляет собой как бы «плохую репутацию» узла или степень его ненадежности. Для того чтобы уменьшить плохую репутацию и завоевать доверие, нужно значительное время решать подзадачи без ошибок. А вот увеличить плохую репутацию можно очень быстро, решив одну-две подзадачи неверно.

Сервер решает, реплицировать ли подзадачу, проверяя следующее условие (по умолчанию $a = 0{,}05$):

$$e(h) > a\ .$$

Если условие выполняется, то узел считается ненадежным и подзадача, выдаваемая ему, реплицируется на другой узел для проверки. Иначе узел считается надежным с вероятностью

$$p_{trust} = 1 - \sqrt{\frac{e(h)}{a}}\ ,$$

и ему выдается нереплицированная подзадача.

При этом достигается гораздо более низкая степень загрузки узлов решением избыточных подзадач, поскольку в целом количество ошибочных результатов из-за аппаратно ненадежных узлов является гораздо меньшим, чем количество подзадач, решаемых при классической репликации (нагрузка

падает, по разным оценкам, с 50 % до 5–10 %). Недостаток данного способа очевиден – из-за оперирования вероятностными терминами приходится смириться с тем, что часть подзадач может быть решена некорректно, и в зависимости от класса решаемой задачи это можно либо допустить, либо необходимы дополнительные проверки после окончания процесса расчетов, а в наихудшем случае – пересчет задачи заново.

Рассмотрим более подробно второй упомянутый нами фактор надежности узла – административный.

Как показывает практика расчетов в распределенных вычислительных системах рассматриваемых типов, именно этот фактор наиболее сильно влияет на их общую производительность. Вычислительные узлы в таких системах имеют тенденцию «исчезать» и «появляться», при этом подзадачи, выданные узлам, естественно, решаются далеко не всегда.

Естественным решением для таких систем, казалось бы, является уменьшение объема выдаваемой узлам подзадач до минимально возможного. При этом потерянная подзадача легко и быстро может быть решена заново. Однако такая стратегия хороша только на первый взгляд. Дело в том, что в этом случае пропорционально уменьшению объема подзадачи растут расходы на приемо-передачу данных по сети и время обработки подзадачи в соответствии с алгоритмом функционирования системы распределенных вычислений (упаковка/распаковка данных, создание процессов расчета, проверка и т.д.). Поэтому просто так уменьшать объем подзадачи нельзя, следует воспользоваться математическими методами расчета объемов подзадач, которые учитывали бы большинство имеющихся факторов.

Для распределенных систем с достаточной степенью статичности состава автором предлагается стратегия повышения производительности с учетом отказов узлов [2]. Под отказом мы, как уже было сказано, понимаем недоступность узла, вызванную как аппаратными, так и иными факторами, в совокупности составляющими «административный отказ».

Данная стратегия опирается на статистику отказов (характеристику надежности узла), полученную опытным путем. Для ее реализации на серверном узле (узлах) размещается специальное программное обеспечение, вычисляющее функцию надежности $P_w(t)$ для каждого узла, основанную на анализе журнала доступности узла за определенный период времени.

Фактически данное программное обеспечение представляет собой в простейшем случае модуль, осуществляющий периодическую процедуру *ping* в отношении каждого узла вычислительной системы. Результат каждого отклика заносится в журнал (лог-файл или таблицу базы данных). Впоследствии анализатор считывает эти данные и формирует характеристику надежности каждого узла.

Чем больше период анализа функционирования узла, тем более точной получается его характеристика надежности (т.е. зависимость вероятности безотказной работы от времени работы). Оператор вычислительной системы имеет возможность просмотреть журнал доступности и графики надежности для каждого узла в удобном виде (рис. 1).

В более сложном случае модуль может высчитывать не только доступность каждого узла в определенный момент времени, но и иные характери-

стики, анализ которых позволит осуществить оптимизацию процесса разбиения задачи. К примеру, можно отмечать для каждого проверочного отсчета такие характеристики, как загрузка CPU и GPU, активность дисковой подсистемы и т.д.

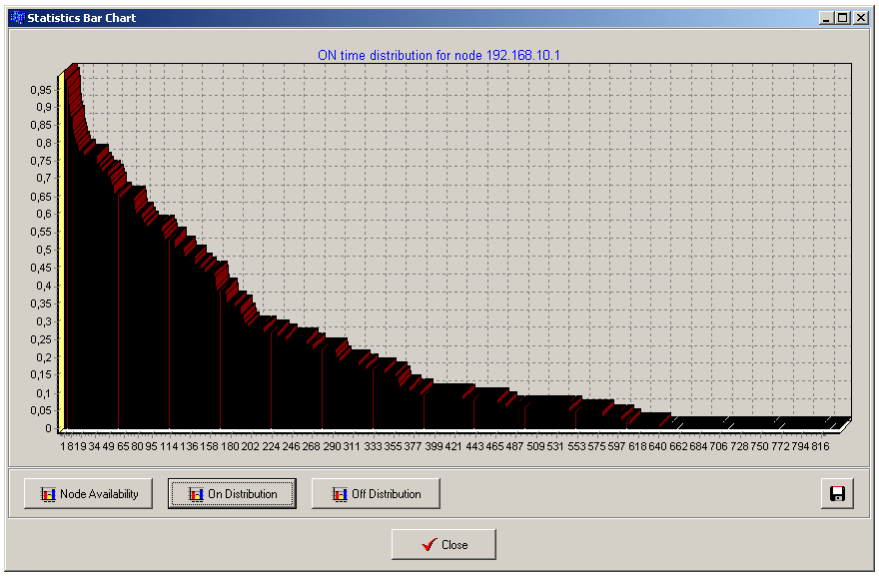

Рис. 1. Статистически полученная функция надежности узла

Стратегия размещения подзадач основана на том факте, что надежность узла постепенно уменьшается со временем его работы, что следует из графика на рис. 1. При этом, чтобы избежать бесполезного расчета подзадачи, которая с той или иной вероятностью будет потеряна во время отказа узла, ему с течением времени функционирования выдаются подзадачи уменьшенного объема, учитывающие время его безотказной работы на каждом этапе. После отказа и восстановления узла считается, что узел опять имеет максимальную надежность, и он получает подзадачу максимального объема. Описание стратегии и формулы для расчета объемов подзадач приведены в [2].

Недостатком предложенной стратегии, как уже было сказано, является то, что она требует наличия статистических данных о функционировании узлов. Такие данные могут быть получены только в системах, имеющих статическую структуру, наличие и состав узлов в которых является постоянным. В реальных системах типа Desktop Grid выполнить это требование достаточно сложно. Поэтому необходима такая стратегия размещения подзадач, которая в динамике учитывала бы изменение состава системы распределенных вычислений и надежностные характеристики ее узлов.

Анализ процесса функционирования узлов показал, что такая стратегия может быть получена с использованием теории марковских случайных процессов.

Одним из определений марковского процесса является следующее утверждение: при фиксированном состоянии процесса в настоящий момент времени будущее и прошлое состояния марковского процесса независимы [3]. Или, можно сказать, что случайный процесс, протекающий в систе-

ме, называется марковским, если для любого момента времени t_0 вероятность любого состояния системы при $t > t_0$ зависит только от ее состояния при $t = t_0$ и не зависит от того, как и когда система пришла в это состояние.

В нашем случае, как показано на рис. 2, процесс функционирования каждого вычислительного узла представляет собой чередование двух состояний (работы и восстановления), причем состояние, в которое узел перейдет в определенный момент, зависит только от того, в каком состоянии он находился до перехода (состояния чередуются), и не зависит от более ранних состояний узла.

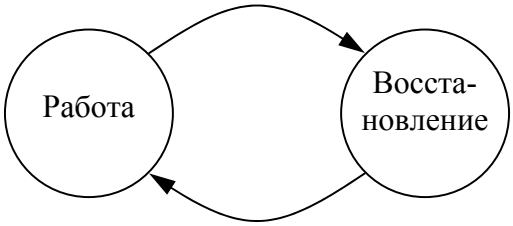

Рис. 2. Возможные состояния вычислительного узла

Таким образом, мы можем рассматривать процесс функционирования каждого вычислительного узла системы как дискретный марковский процесс с двумя состояниями.

Процесс $\Theta(t)$ в любой момент времени может иметь лишь одно из значений $v_1 = 1$ (работа) и $v_2 = 0$ (восстановление), причем вероятность перехода $v_1 \to v_2$ (отказ узла) за малое время Δt равна $\lambda \Delta t$, а вероятность перехода $v_2 \to v_1$ (возврат узла к работе) равна $\mu \Delta t$. Известны вероятности начального состояния $p_1^0 = 1$, $p_2^0 = 0$.

Имея эти исходные данные, можно определить вероятность перехода $\pi_{ij}(t_0, t) = P\{\Theta(t) = v_j \,|\, \Theta(t_0) = v_i\}$, где $v_1 = 1$, $v_2 = 0$, $i, j = 1,2$.

В общем случае имеет место система линейных дифференциальных уравнений, полученных из уравнения Колмогорова – Чепмена:

$$\frac{\partial}{\partial t}\pi_{ij}(t_o, t) = \sum_{k=1}^{2} a_{kj}(t)\pi_{ik}(t_0, t), \; i, j = 1,2,$$

где a_{kj} – крутизна изменения вероятности на небольшом отрезке времени.

Для того чтобы применительно к реальным условиям иметь возможность решить эту систему уравнений, необходимо некоторое упрощение, в качестве которого и приняты ранее указанные соотношения $a_{12} = \lambda$, $a_{21} = \mu$, и из условий нормировки $a_{11} = -\lambda$, $a_{22} = -\mu$.

Это значит, что на достаточно небольшом отрезке времени график функции надежности узла можно аппроксимировать линейной функцией.

Поэтому воспользуемся методикой расчета переходных вероятностей для дискретного марковского процесса с двумя состояниями и с постоянными

коэффициентами a_{ij}, предложенной в [3]. Исходя из этой методики в результате решения системы уравнений, полученной из уравнений Колмогорова – Чепмена, с учетом постоянных коэффициентов получим следующие выражения для переходных вероятностей:

$$\pi_{11}(\tau)=\frac{\mu}{\lambda+\mu}+\left(\frac{\lambda}{\lambda+\mu}\right)e^{-(\lambda+\mu)\tau},\ \pi_{12}(\tau)=\left(\frac{\lambda}{\lambda+\mu}\right)\left[1-e^{-(\lambda+\mu)\tau}\right],$$

$$\pi_{22}(\tau)=\frac{\lambda}{\lambda+\mu}+\left(\frac{\mu}{\lambda+\mu}\right)e^{-(\lambda+\mu)\tau},\ \pi_{22}(\tau)=\left(\frac{\mu}{\lambda+\mu}\right)\left[1-e^{-(\lambda+\mu)\tau}\right].$$

Прежде всего нас из этих вероятностей интересует вероятность перехода π_{12} узла из состояния 1 в состояние 2, т.е. отказ узла.

Определим значение времени τ из этого выражения:

$$\tau=-\frac{1}{\lambda+\mu}\ln\left[1-\frac{\pi_{12}(\lambda+\mu)}{\lambda}\right].$$

Полученное выражение позволит нам, задавшись вероятностью отказа узла $\pi_{12}<\frac{\lambda}{\lambda+\mu}$, получить время этого отказа τ с учетом поведения узла, описываемого функцией надежности.

То есть фактически мы получаем возможность определить время безотказной работы узла на каждом этапе. А зная время безотказной работы узла, а также его производительность и пропускную способность канала связи, мы легко можем выдать ему подзадачу такого объема, чтобы она была решена в рамках обозначенного времени.

Перед началом работы необходимо осуществить начальное разбиение задачи. Метод этого разбиения может быть любым. Можно предложить два простых варианта:

1) равномерное разбиение;

2) разбиение с учетом производительности узлов.

При этом по мере работы изначально выданный объем данных будет корректироваться и приближаться к значению, которое можно охарактеризовать как «максимальный объем подзадачи за время безотказной работы узла».

Общий алгоритм работы сервера в данном случае будет таким.

После выдачи стартового набора подзадач вычислительным узлам сервер переходит в состояние приема результатов. Для каждого узла, приславшего результат, сервер отмечает время его расчета и выдает следующую подзадачу.

Если узел вместо результата решения подзадачи присылает серверу пустой запрос на решение, сервер по этому признаку определяет, что имел место отказ узла во время решения ранее выданной ему подзадачи. К этому моменту серверу известны следующие характеристики, относящиеся к этому узлу:

– количество выданных узлу и успешно решенных подзадач K;

– объем каждой подзадачи S_j;

– время решения каждой подзадачи узлом t_j;

– время, прошедшее с момента выдачи последней подзадачи до момента пустого запроса узла на выдачу новой подзадачи t_{K+1}.

Из этих характеристик можно приблизительно определить время безотказной работы узла на первом этапе:

$$t_w^{(1)} \approx \sum_{j=1}^{K} t_j ,$$

и время восстановления после первого отказа:

$$t_f^{(1)} \approx t_{K+1} .$$

Эти времена позволят приближенно определить параметры линейных функций, аппроксимирующих функцию надежности и функцию восстановления. Эти параметры определяются как интенсивности переходов:

$$\lambda^{(1)} = \frac{1}{t_w^{(1)}}, \ \mu^{(1)} = \frac{1}{t_f^{(1)}} .$$

Имея эти коэффициенты, мы определяем значение времени безотказной работы узла τ.

Дальнейшее распределение подзадач осуществляется уже с учетом найденного значения времени τ так, чтобы данный узел получил подзадачу такого объема, который будет рассчитан в пределах промежутка времени $(0;\tau)$. После следующего отказа узла процесс перерасчета повторяется.

Коррекция на каждом шаге значений λ и μ приводит к тому, что по мере протекания процесса вычислений и по мере накопления сведений о длительности промежутков работы и восстановления каждого узла выдаваемые узлам объемы подзадач все точнее приближаются к оптимальному значению, которое можно выразить как максимальный объем за время безотказной работы узла.

Таким образом, предложенная стратегия размещения подзадач позволит уменьшить влияние административных отказов на протекание процесса вычислений в распределенной системе, а также уменьшить нагрузку на сеть и на узлы и, таким образом, увеличить производительность вычислений.

Библиографический список

1. Программное обеспечение с открытым исходным кодом для организации добровольных распределенных вычислений и распределенных вычислений в сети. – URL: http://boinc.berkeley.edu/
2. **Токарев, А. Н.** Оптимизация размещения подзадач в распределенных вычислительных системах с ненадежными узлами / А. Н. Токарев // Актуальные проблемы современной науки : тр. 1-го Междунар. форума. – Самара, 2005. – С. 113–116.
3. **Горяинов, В. Т.** Статистическая радиотехника : примеры и задачи / В. Т. Горяинов, А. Г. Журавлев, В. И. Тихонов. – М. : Сов. радио, 1980. – 544 с.

References

1. *Programmnoe obespechenie s otkrytym iskhodnym kodom dlya organizatsii dobrovol'nykh raspredelennykh vychisleniy i raspredelennykh vychisleniy v seti* [Open source

software for organizing voluntary computing and distributed computing in a network]. Available at: http://boinc.berkeley.edu/
2. Tokarev A. N. *Aktual'nye problemy sovremennoy nauki: tr. 1-go Mezhdunar. Foruma* [Actual problems of modern science: proceedings of the 1st International Forum]. Samara, 2005, pp. 113–116.
3. Goryainov V. T., Zhuravlev A. G., Tikhonov V. I. *Statisticheskaya radiotekhnika: primery i zadachi* [Statistical radio engineering: examples and tasks]. Moscow: Sov. radio, 1980, 544 p.

Токарев Андрей Николаевич
магистрант, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: ant19@mail.ru

Tokarev Andrey Nikolaevich
Master's degree student, Penza State University (40 Krasnaya street, Penza, Russia)

УДК 004.75
Токарев, А. Н.
Уменьшение нагрузки и повышение производительности вычислений в распределенных вычислительных системах с ненадежными узлами / А. Н. Токарев // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2018. – № 2 (46). – С. 37–47. – DOI 10.21685/2072-3059-2018-2-4.