УДК 681.3.06+519.68

ПРАКТИЧЕСКИЙ ПОДХОД К ВЗАИМОДЕЙСТВИЮ СИСТЕМ МОНИТОРИНГА ВЫЧИСЛИТЕЛЬНЫХ КЛАСТЕРОВ СО СТОРОННИМ ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ

А. Г. Тарасов[1]

Приводятся практические примеры осуществления автоматизированного контроля ресурсов вычислительного комплекса. Описывается применение подхода к взаимодействию системы мониторинга со сторонним программным обеспечением посредством уведомлений о событиях. Работа выполнена в рамках ФЦП “Научные и научно-педагогические кадры инновационной России” (проект № 02.740.11.0626) и поддержана грантом ДВО РАН № 09–I–П1–01.

Ключевые слова: архитектура систем мониторинга, вычислительные кластеры, искусственные нейронные сети.

1. Введение. При вводе в эксплуатацию вычислительного комплекса (ВК) разработчикам необходимо решить ряд задач. Одной из них является развертывание средств управления, а также систем мониторинга (СМ). Традиционно для исследования вопросов, связанных с организацией сбора данных мониторинга и контроля ВК, статистики использования локальных вычислительных сетей и программных комплексов разрабатывалось прикладное программное обеспечение (ПО). Исследованиям в этой области посвящены работы [1–3] и др. Значительные результаты были достигнуты и в смежных областях: в теории управления и управляющих систем, теории автоматов, системном анализе и системах поддержки принятия решений [4–7].

Тем не менее, задачи мониторинга по-прежнему являются сложными в практической реализации. Контроль компонентов программно-аппаратного комплекса и оказание управляющих воздействий являются критически важными для организации высокопроизводительных распределенных вычислений. Пользователь и администратор нуждаются в информации о том, как выполняется отправленное на ВК задание (под заданием в тексте понимается приложение и описание ресурсов, необходимых ему для выполнения), какое влияние оно оказывает на вычислительную систему. При этом с ростом числа вычислительных узлов в составе ВК возрастает и объем данных, которые приходится анализировать администратору ВК, увеличивается вероятность ошибки вследствие человеческого фактора.

Многие СМ успешно и эффективно решают задачи мониторинга в частных случаях. Тем не менее, для ВК, на которых установлено устаревшее ПО мониторинга, зачастую переход на новые системы мониторинга затруднен, а устаревшее ПО уже не удовлетворяет возросшим требованиям. Несмотря на все многообразие архитектур СМ, до сих пор отсутствует возможность интеграции различных СМ между собой, затруднено добавление возможностей к уже разработанным СМ. Поэтому существует потребность в разработке новых подходов к архитектуре СМ, позволяющих устранить указанные выше недостатки.

В [8] автором была описана расширяемая архитектура СМ и созданное на ее основе приложение Grate, предоставляющее набор служб, позволяющих устранить описанные выше сложности. Разработанная архитектура позволяет наращивать возможности мониторинга, в том числе добавлять сторонние модули к уже используемым СМ, не останавливая их работу, а также осуществлять совместный мониторинг и анализ данных, поступающих от различных СМ. В настоящей статье основное внимание уделяется вопросам взаимодействия СМ на базе разработанной архитектуры с приложениями как для расширения их функциональных возможностей, так и для добавления функций, отсутствующих в базовой реализации СМ Grate.

Разработка и применение рассматриваемых в статье программных средств выполнялись в рамках проекта по созданию распределенной вычислительной среды (РВС) ВЦ ДВО РАН.

2. Задачи систем мониторинга и подход к их решению. Сформулируем основные задачи мониторинга. Это — получение от вычислительного комплекса и сохранение значений контролируемых характеристик; выявление значимых событий на основе полученных значений; запуск средств реагирования (отклика) на значимое событие.

[1] Вычислительный центр ДВО РАН (ВЦ ДВО РАН), ул. Ким-Ю-Чена, 65, 680000, Хабаровск; науч. сотр., e-mail: taleks@as.khb.ru

Значимым событием назовем переход вычислительной системы в такое состояние, при котором происходит существенное для контролируемых характеристик изменение значений, указывающее на важные для пользователя СМ изменения в работе ВК. К таким состояниям можно отнести недостаток оперативной памяти на вычислительном узле, аномально высокая активность работы подсистемы ввода/вывода и др. Контролируемыми величинами обычно являются степень использования оперативной памяти вычислительного узла, объем свободной дисковой памяти, степень использования центрального процессорного устройства (ЦПУ) и др.

Для решения перечисленных выше задач была предложена архитектура СМ, передача данных в которой может осуществляться лишь между соседними логическими уровнями СМ [9, 10]. Это позволило изменять, дополнять и расширять уровни, не нарушая общей работоспособности СМ. Одним из важных следствий такого подхода стало облегчение взаимодействия СМ со сторонним ПО, что, в свою очередь, позволяет расширять функциональные возможности мониторинга.

Важными элементами предложенной архитектуры являются триггеры.

Триггер — это логический элемент, содержащий условие или выражение, истинность которого необходимо проверить для последнего значения контролируемой характеристики (*метрики*). Если условие выполнено, то состояние триггера изменяется, что приводит к выполнению определенных при создании триггера действий.

Каждый контролируемый компонент в СМ может генерировать уведомление о возникшем событии. Применение механизма подписки на уведомления о событиях обеспечивает взаимодействие СМ со сторонним ПО. Приложение пользователя может обрабатывать уведомления о событиях, отправляемых контролируемыми узлами СМ, игнорировать их или же генерировать новые события, которые будут обработаны другими подписчиками.

В численных экспериментах, приведенных ниже, применяется разработанная СМ с использованием архитектуры Grate, состоящей из нескольких независимо функционирующих модулей, написанных на языке программирования Java. Эти модули используют общую библиотеку интерфейсов и классов, позволяющую им работать с метриками, триггерами, событиями и прочими элементами СМ.

3. Расширение функциональных возможностей систем мониторинга. Разработанная СМ Grate позволяет организовывать взаимодействие с ПО одним из двух способов:

— расширять функциональные возможности сторонних СМ,

— упрощать обмен данными и уведомлениями о событиях между различными модулями посредством служб СМ.

Например, на кластере КВЦ-1, собранном в 2004 г. из неспециализированных компонентов, для контроля характеристик ВК использовалась СМ Ganglia [11]. Невозможность создания отклика системы штатными средствами используемой СМ привела к необходимости добавления при помощи СМ Grate возможности контроля негативных изменений повышения температуры вычислительных узлов, например в случае отказа системы активного охлаждения.

Интеграция возможностей разработанной СМ выполнялась следующим образом: в модуле grated была реализована бóльшая часть функциональности службы gmetad СМ Ganglia, данные мониторинга поступали от служб gmond с предварительно настроенной метрикой температуры. Эта метрика не является стандартной для СМ Ganglia, но ее возможно формировать с использованием стандартных служб ОС и передавать через приложение gmetric, входящее в состав СМ Ganglia.

При этом проводилась проверка триггеров, были также доступны и остальные возможности разработанной СМ. При превышении заданной величины происходила запись в журнал событий и отправка уведомления администратору кластера по электронной почте. В критическом случае принудительное отключение узла инициировалось аппаратными средствами, однако сигнал о необходимости отключения был использован для подготовки программного обеспечения к остановке работы и отключению питания. Модуль grated проводил проверку триггеров (по одному на каждый вычислительный узел), расширяя тем самым функциональные возможности используемой СМ Ganglia. Визуализация выполнялась ПО grate-client на основе данных, поставляемых от grated, однако, тем не менее, веб-фронтэнд СМ Ganglia в штатном режиме мог обрабатывать и предоставлять доступ к данным, получаемым от grated.

Важным результатом является расширение возможностей установленной системы без остановки ее работы, без перенастройки источников данных. Замена агента сбора данных позволила получить сравнимую или лучшую производительность при более широких функциональных возможностях. Получить такой результат с использованием других доступных на момент тестирования СМ было невозможно.

Тестирование, выполненное при мониторинге до 15 узлов ВК, показало, что производительность СМ Grate (модуля grated) выше, чем у СМ Ganglia (модуля gmetad). Тем не менее, grated использует боль-

шие объемы оперативной памяти на управляющем сервере в силу использования виртуальной машины Java [10]. В настоящее время оценить рост затрат на реальном кластере с числом узлов более 15 не представляется возможным из-за отсутствия у автора доступа к подобным ВК для получения данных мониторинга. Тем не менее, можно оценить рост затрат производительности исходя из имеющихся фактических данных и результатов моделирования контроля 100 вычислительных узлов по методике оценки масштабирования, использованной разработчиками СМ Ganglia.

При использовании источников данных, аналогичных СМ Ganglia (gmond), и групповой передачи данных (multicast, протокол UDP) рост накладных расходов на организацию сетевых подключений для получения значений контролируемых характеристик с увеличением числа узлов незначителен. Большее влияние на рост вычислительных затрат оказывает проверка условий триггеров, общее количество которых зависит от числа вычислительных узлов. При проведении численных экспериментов данное утверждение подтверждается: для 8 узлов выигрыш в производительности grated в сравнении с gmetad составляет 45%, при увеличении числа узлов разница уменьшается по закону, близкому к линейному, доходя при моделировании мониторинга 100 вычислительных узлов до 10%. Максимальные затраты на мониторинг составляют менее 0.5% от всего времени работы ЦПУ сервера, на котором запускался grated (по данным за 41 день статистики использования вычислительного времени процессом java, который исполнял grated).

Тот же подход к расширению функциональных возможностей СМ можно применить в задачах управления потреблением питания ВК. Серверное оборудование в основном сконфигурировано для достижения максимальной производительности в вычислительных задачах, поэтому даже в режиме простоя потребление энергии зачастую незначительно меньше пикового потребления при полной нагрузке во время вычислительного процесса. Принудительное отключение питания неиспользуемых узлов в кластере может позволить сократить использование электроэнергии “вхолостую”.

Схожей задачей является и уменьшение потребления энергии за счет снижения частоты центральных процессоров и перевода ряда устройств в режим сна (режим пониженного потребления электропитания). Это особенно важно при нарушениях электроснабжения. Потребляемая ВК мощность не позволяет блокам бесперебойного питания обеспечивать длительную работу вычислительных узлов; в то же время, прерывание длительного вычислительного процесса недопустимо. Снижение потребляемой мощности позволяет ВК дольше работать от батарей питания и подготовиться ко включению резервных генераторов электроэнергии. Использование СМ с поддержкой триггеров позволяет упростить решение подобных задач.

Другой важной задачей является выявление значимых событий, что позволяет более полно использовать ВК за счет сокращения не являющихся полезными затрат вычислительных ресурсов. Поскольку с ростом числа узлов и контролируемых характеристик системному администратору становится все сложнее в непрерывном режиме следить за изменениями в значениях опрашиваемых характеристик ВК, то возникает потребность в автоматизации рутинных операций. При этом многие изменения в работе ВК могут быть обнаружены и в ряде случаев предотвращены с использованием различных систем автоматизированного контроля.

Примером таких обнаруживаемых изменений могут служить последствия неэффективно работающего приложения, т.е. не использующего ресурсы ВК полностью во время своего выполнения. Например, довольно типичной ситуацией является неверный выбор семейства функций (групповая рассылка или рассылка точка-точка, блокирующие или неблокирующие операции отправки данных) при пересылке MPI-сообщений в программе пользователя, а также простаивание на операциях ввода-вывода дисковой подсистемы, что выявляется по относительно большой доле времени работы ЦПУ в пространстве ядра (полезные вычисления программ пользователя производятся ЦПУ в пространстве пользователя).

На рис. 1 изображены данные мониторинга вычислительного узла, на котором приложение длительное время занималось в основном работой с файлами, полезные вычисления составляли менее 30% процессорного времени (область A на графике). Такая ситуация хорошо распознается обученной искусственной нейронной сетью (ИНС) и может быть предотвращена при своевременном вмешательстве администратора ВК или разработчика программы. Например, после рекомендаций по изменению размеров блоков данных при использовании системных вызовов ОС приложение пользователя, запущенное позднее (область B), было более высокопроизводительно.

В приведенном примере задание не нарушало работы приложений других пользователей и не представляло опасности для функционирования ВК, однако в работе кластера ВЦ ДВО РАН произошла однажды и более серьезная проблема с использованием одним из заданий виртуальной памяти в объеме, приведшем к существенному использованию файла подкачки и, вследствие этого, отказу ряда системных приложений, нуждавшихся в свободной оперативной памяти. Сопутствующим результатом стало общее замедление работы всех приложений, разделяющих узел ВК со сбойным вычислительным заданием. Оче-

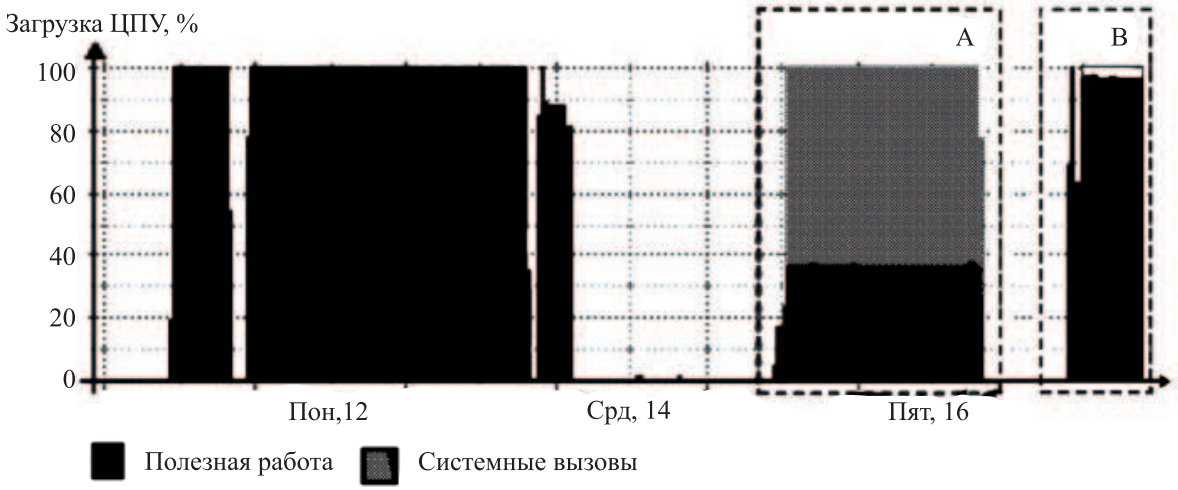

Рис. 1. График неэффективного использования ресурсов приложением пользователя

видно, такое поведение приложения недопустимо, критическая ситуация должна быть предотвращена как можно раньше.

В [12, 13] рассмотрена возможность применения ИНС нескольких типов в целях выявления состояния, в котором находится вычислительный узел и ВК в целом. Было показано, что обученная ИНС встречного распространения способна верно определять текущее состояние ВК при заранее не известных входных данных. Это обусловило выбор модуля ИНС в качестве компонента разработанной СМ, ответственного за выявление изменений в работе ВК.

Схема взаимодействия данной СМ и модуля ИНС, который выполняется как независимое приложение, приведена на рис. 2. Стрелки отображают направления возможной передачи данных и управляющих воздействий. Тонкими стрелками показаны направления постоянно существующих потоков данных. Таким образом, в эксперименте участвовали четыре компонента: вычислительный кластер, система мониторинга, приложение-обработчик событий и модуль ИНС.

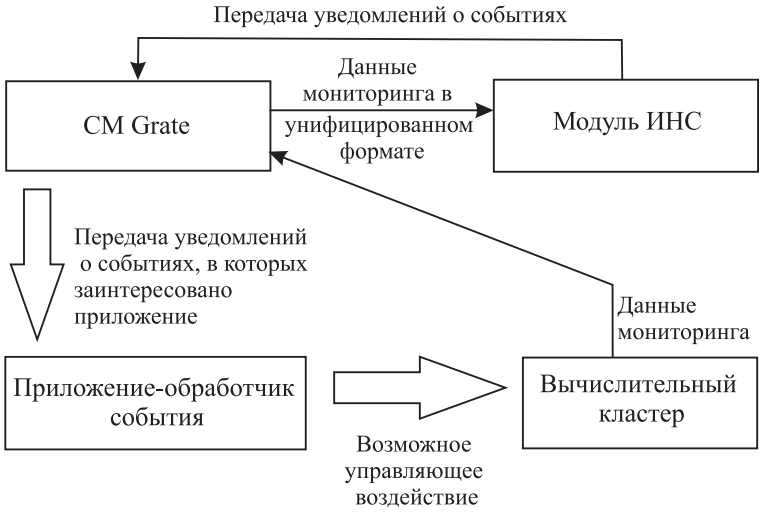

Рис. 2. Схема взаимодействия модуля ИНС и CM Grate

Вычислительной платформой эксперимента служил кластер, состоящий из 4 узлов под управлением операционной системы CentOS Linux v.4.4. Модуль ИНС выполнялся как независимое приложение, получающее данные от службы grated. После обработки входных данных система мониторинга получала информацию о событиях (изменениях в состоянии ВК) через механизм уведомлений. Приложение-обработчик получало информацию о событии, выполняло запись в журнал и посылало электронную почту администратору ВК. Приложение-обработчик и служба grated CM Grate выполнялись на управляющем узле, модуль ИНС — на отдельной вычислительной станции.

Было проведено тестирование корректности функционирования модуля ИНС, взаимодействующего с разработанной СМ посредством уведомлений. Результаты работы модуля ИНС в режиме без взаимодействия с СМ были известны из ранее проведенных исследований и использовались в качестве эталонных.

В процессе предварительного исследования были выделены следующие значимые входные параметры для работы ИНС:

— доля свободного места в файле подкачки,

— отношение числа процессов к числу ядер,

— доля свободной оперативной памяти,
— отношение загруженности процессора пользовательскими процессами к системным процессам,
— доля свободной памяти на жестком диске,
— число принятых байт на сетевых интерфейсах,
— число отданных байт,
— производные от них величины.

Такие параметры измерялись как для кластера в целом, так и для каждого узла в отдельности.

При обучении ИНС в разработанном ПО возможно варьировать следующие параметры:
— скорость обучения,
— необходимая точность (если проводится обучение с проверкой),
— максимальное число эпох (итераций обучения).

В результате обучения методом обратного распространения ошибки была получена двухслойная ИНС, которая способна на основе входных данных для одного вычислительного узла (или для ВК) выдать на выходах значения, указывающие на вероятность нахождения узла (или ВК) в одном из состояний. Под состоянием понимаются значения различных значимых метрик (температура процессора, количество свободной оперативной памяти, число выполняемых процессов и др.). В процессе работы модуль ИНС также запоминал входные параметры, при которых ИНС давала неопределенный ответ, что позволяло в дальнейшем построить на этих данных обучающую выборку и корректировать ИНС.

В процессе исследования была выбрана и испытана ИНС встречного распространения с сигмоидальной функцией активации. Более подробную информацию о функционировании ИНС и методиках формирования входных векторов данных, в том числе о нормализации входных векторов данных и задании случайных начальных ненулевых весов, можно получить в [14, 15].

Для получения информативной выборки входных параметров необходим большой объем данных по состоянию узлов во время штатной работы ВК. С целью накопления достаточного количества входных данных на экспериментальном вычислительном кластере был выполнен тест HPL.

Модуль ИНС опрашивал состояние ВК через систему мониторинга. При этом сеть обучалась для выявления следующих основных классов состояний:

— “неэффективное использование ЦПУ” (в смысле малой доли полезных вычислений в пространстве пользователя; это состояние было смоделировано специально разработанной утилитой, поскольку получить его из данных мониторинга теста HPL было невозможно);

— “штатная работа” (доля полезных вычислений до 75%, обмен данными между вычислительными узлами невелик);

— “стадия интенсивных вычислений” (доля полезных вычислений более 75%, значительный обмен данными между узлами ВК);

— “простой ВК” (нагрузка ЦПУ на контролируемых узлах составляет менее 5%).

В рамках исследования взаимодействия нейросетевого модуля с СМ было выполнено более ста тестов HPL с различными параметрами по методике, указанной в [12], с единственным отличием: данные мониторинга предоставлялись не выборкой из базы данных RRD (Round-Robin Database) СМ Ganglia, а из разработанной СМ в реальном времени. Для автоматизации тестирования был создан shell-скрипт, который:

1) формировал файл описания задания (тест HPL),
2) ставил задание на выполнение через систему диспетчеризации PBS Torque,
3) для каждого вычислительного узла извлекал необходимые данные из полученных от СМ Grate,
4) сохранял результат в файл для дальнейшего анализа корректности работы модуля ИНС.

Было проведено обучение ИНС при различных комбинациях входных векторов. Результаты работы сети для неизвестных на момент обучения наборов данных до 2–3 знаков совпадают с приведенными в работе [12], что говорит о корректности взаимодействия СМ и модуля ИНС и применимости такого взаимодействия для выявления значимых событий в реальном времени.

Затем на период тестирования (17 дней) модуль ИНС был запущен в режиме контроля экспериментального ВК, было проверено более 4000 входных векторов для каждого из узлов. Для подтверждения повторяемости результатов было проведено несколько численных экспериментов, повторяющих методику обучения, при этом обучающая выборка, общее время эксперимента и объем участвующих в проверке данных были меньше, но результаты в численном выражении оказались близки приведенным ниже.

Имея на входе значения метрик, ИНС встречного распространения при 12 нейронах и 30 циклах обучения верно отнесла текущее состояние к одному из классов состояний, приведенных выше. При изменении класса состояния модуль ИНС посылал уведомление о событии в СМ, которая, в свою очередь,

передавала его приложению-обработчику. Была также смоделирована ситуация активного использования файла подкачки с помощью тестовой задачи, активно выделяющей оперативную память без возврата ее системе. ИНС успешно распознала критическую ситуацию и уведомила администратора кластера.

В результате проведения численных экспериментов с модулем ИНС выяснилось, что сети с большим числом нейронов требуют увеличения количества циклов обучения для достижения корректного распознания тестовых входных векторов (30 циклов для ИНС из 12 нейронов, около 120 для 20 нейронов, свыше 300 для 50 нейронов). С ростом количества циклов обучения или числа наборов данных, подаваемых на вход ИНС, со временем может возникнуть ситуация нехватки вычислительных ресурсов для работы ИНС в реальном времени. В связи с этим было проведено тестирование вычислений на графических процессорах (ГПУ) с использованием технологии OpenCL (http://www.khronos.org/opencl/).

В тестировании (рис. 3) в качестве сервера мониторинга участвовали две различные конфигурации:

1) MacOS X 10.6.4 x86_64, nVidia 9400M, общая память, Intel Core 2 Duo P7350, OpenCL 1.0;

2) Linux CentOS 5.5 x86_64, nVidia GTX 285, OpenCL 1.1.

В случае сервера с конфигурацией MacOS X драйвера позволяют проводить вычисление *ядер* (kernel в терминах стандарта OpenCL) на ЦПУ, драйвера в ОС Linux такой возможности на момент тестирования не имели. Представленное на графике время включает только запуск экземпляров ядер в очереди OpenCL, значения оси абсцисс соответствуют различным конфигурациям ИНС (число входов, число нейронов первого слоя, число нейронов второго слоя). При этом все входные данные при запуске приложения единожды передавались в область памяти, доступную ядрам OpenCL. Такой подход позволяет исключить влияние операций ввода-вывода на время расчетов (на практике все данные по ВК получить единовременно невозможно, поскольку они порционно поступают в СМ и накапливаются в течение периода времени, который зависит от набора контролируемых характеристик).

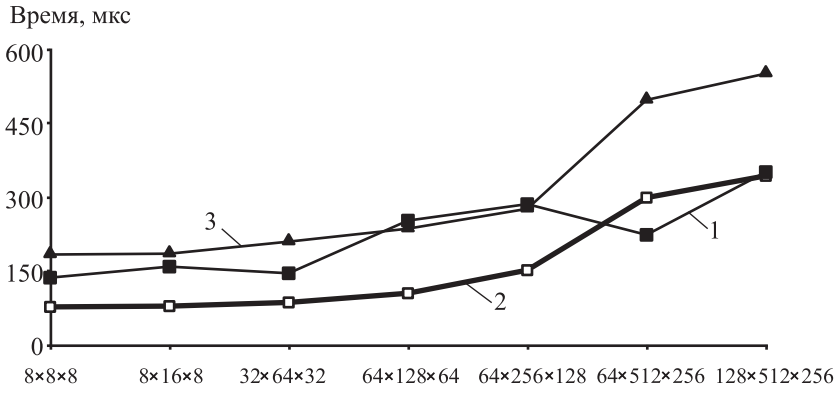

Рис. 3. Время работы ИНС для одного набора входных данных: 1) nVidia 9400M, 2) Intel P7350, 3) nVidia GTX285

Для оптимизации вычислений в ядрах были использованы операции над 4-мерными векторами (в результате чего необходимо использовать число входов и число нейронов, кратные 4), вычисления производились над вещественными числами одинарной точности. На графике приведено среднее значение времени прогона входного вектора для каждого типа ИНС на основе 30 испытаний.

Для сравнения в авторской реализации модуля ИНС на языке Java одна итерация обучения ИНС на базе выборки из 470 входных векторов занимала 80–100 мс на процессоре Intel Core 2 Duo E8500 (3.16 ГГц) для следующей конфигурации ИНС: 21 вход, 25 нейронов первого слоя, 25 нейронов второго слоя, 5 выходов (равное числу выявляемых состояний). Один прогон такой ИНС занимал 140–220 мкс, при этом рост затрат производительности модуля ИНС пропорционален количеству контролируемых узлов, поскольку один прогон входных данных ИНС соответствует обработке данных мониторинга для одного вычислительного узла (или ВК). Объем данных, передаваемых по сети от СМ до модуля ИНС, также пропорционален числу узлов, но является незначительным даже для крупных ВК (значения всех типичных метрик для вычислительного узла в конкретный момент времени описываются пакетом данных, размер которого менее 2 Кб в текстовом представлении и менее 300 байт в двоичном формате).

Отметим, что преимуществ в скорости вычисления ИНС у ГПУ не наблюдается вплоть до конфигурации, которая мало применима на практике при мониторинге ВК. Это обусловлено тем, что ядро, реализующее нейрон, довольно простое и является сумматором с сигмоидальной функцией активации. Такие ядра с высокой эффективностью вычисляются и на ЦПУ. Использование технологии OpenCL для небольших нейронных сетей также затруднено большими накладными расходами на передачу данных из ОЗУ в видеопамять и сопутствующую вычислениям работу драйверов.

На рис. 4 приведены результаты тестирования ИНС по обработке 5000 наборов входных данных, по оси ординат отложено общее время работы программы, от старта до завершения. Для каждой конфигурации ИНС было проведено 30 испытаний, на графике отложено среднее значение, максимальное

отклонение от среднего значения времени работы программы в каждом испытании не превышает 7%.

Несмотря на относительно небольшое время, затрачиваемое для прогона одного входного вектора, в наших задачах мониторинга ВК использование участвовавших в тестировании ГПУ не является оправданным при наличии свободных вычислительных ресурсов ЦПУ. Производительности исследуемого модуля ИНС было достаточно для обработки 500–1000 входных векторов в секунду, включая накладные расходы на получение данных мониторинга, формирование уведомлений о событии смены состояния и прочие вычислительные затраты, не связанные с расчетами ИНС. Такая производительность ПО позволяет осуществлять в реальном времени мониторинг порядка пятисот узлов ВК, если все значения метрик одного узла приходят не чаще, чем раз в секунду (что согласуется с практикой: обычно СМ настроены так, чтобы получать значения метрик не чаще, чем раз в 30–60 секунд).

В тестировании нами для определения изменений в работе ВК в автоматизированном режиме использовались ИНС с относительно несложной топологией. Тем не менее, развитие модуля ИНС позволяет изменять тип и конфигурацию ИНС, не переписывая исходного кода CM Grate. Уведомления о событиях и данные мониторинга могут обрабатываться одновременно несколькими приложениями (или, например, независимыми модулями ИНС), что позволяет более полно анализировать состояние ВК с применением в том числе и других методов и алгоритмов.

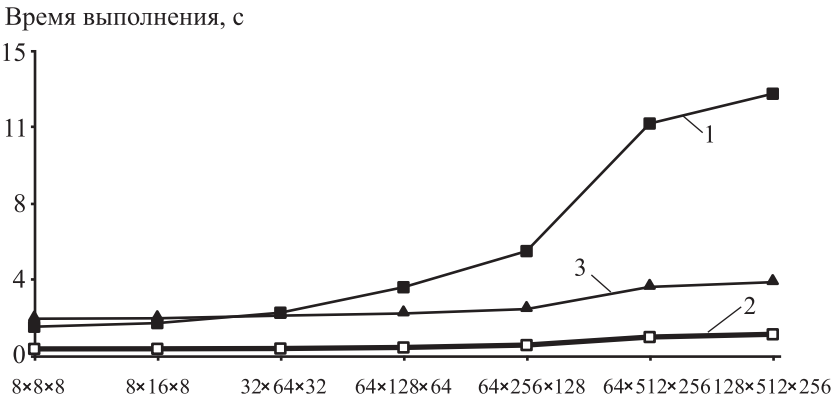

Рис. 4. Общее время выполнения модуля ИНС для 5000 входных векторов: 1) nVidia 9400M, 2) Intel P7350, 3) nVidia GTX285

4. Заключение. В последние годы к системам мониторинга предъявляется все больше требований. Не всегда эти требования можно удовлетворить с использованием возможностей устаревшего программного обеспечения мониторинга. Способность расширения доступных возможностей СМ является важной для автоматизации операций по выявлению значимых событий в работе ВК. Рост вычислительных затрат на сбор и обработку данных мониторинга также требует создания более совершенных с этой точки зрения СМ.

В настоящей статье приведены результаты экспериментов, показывающие, что предложенный ранее подход к архитектуре СМ позволяет успешно решать задачи мониторинга, осуществляя взаимодействие различных программных комплексов между собой. Использование в процессе работы СМ механизма уведомлений о событиях позволяет улучшать ее функциональные возможности, в том числе путем замены модулей, ответственных за различные аспекты анализа данных мониторинга.

СПИСОК ЛИТЕРАТУРЫ

1. *Wolski R., Spring N.T., Hayes J.* The network weather service: a distributed resource performance forecasting service for metacomputing // J. of Future Generation Computing Systems. 1999. **15**, N 5-6. 757–768.
2. *Foster I., Frey J., Tannenbaum T., Livny M., Tuecke S.* Condor-G: a computation management agent for multi-institutional grids // J. on Cluster Computing. 2002. **5**. 237–246.
3. *Genesereth M.R., Ketchpel S.P.* Software agents // Communications of the ACM. 1994. **37**, N 7. 48–54.
4. *Ларичев О.И., Петровский А.Б.* Системы поддержки принятия решений: современное состояние и перспективы развития // Итоги науки и техники. Т. 21. М.: ВИНИТИ. 1987. 131–164.
5. *Ляпунов А.А.* О некоторых вопросах обучения автоматов // Принципы построения самообучающихся систем. Киев, 1962. 115–118.
6. *Моисеев Н.Н.* Математические задачи системного анализа. М.: Наука, 1981.
7. *Яблонский С.В.* Некоторые вопросы надежности и контроля управляющих систем // Математические вопросы кибернетики. Вып. 1. М.: Наука, 1988. 5–25.
8. *Тарасов А.Г.* Расширяемая система мониторинга вычислительного кластера // Вычислительные методы и программирование. 2009. **10**, № 1. 147–158.
9. *Тарасов А.Г.* Трехуровневая система мониторинга расширенной функциональности // Параллельные вычислительные технологии. Челябинск, 2008. 464–469.
10. *Пересветов В.В., Сапронов А.Ю., Тарасов А.Г., Шаповалов Т.С.* Удаленный доступ к вычислительному кла-

стеру ВЦ ДВО РАН // Вычислительные технологии. **11**, № 1. 2006. 45–51.
11. *Пересветов В.В., Сапронов А.Ю., Тарасов А.Г.* Вычислительный кластер бездисковых рабочих станций. Препринт ВЦ ДВО РАН № 83. Хабаровск, 2005.
12. *Пересветов В.В., Писарев А.В.* Нейросетевые компоненты мониторинга вычислительного кластера // Информационные и коммуникационные технологии в образовании и научной деятельности. Хабаровск, 2008. 319–324.
13. *Tarasov A.G.* Integration of computing cluster monitoring system // Proc. of the First Russia and Pacific Conference on Computer Technology and Applications (RPC 2010). Vladivostok: IACP FEB RAS. 2010. 221–224.
14. *Круглов В.В., Борисов В.В.* Искусственные нейронные сети. Теория и практика. М.: Горячая линия-Телеком, 2002.
15. *Хайкин С.* Нейронные сети: полный курс. М.: Вильямс, 2006.

Поступила в редакцию
11.11.2010