Инструментальный комплекс метамониторинга распределенных вычислительных сред

И.А. Сидоров, Г.А. Опарин, В.В. Скоров

Федеральное государственное бюджетное учреждение науки
Институт динамики систем и теории управления СО РАН

Работа посвящена решению проблемы мониторинга распределенной вычислительной среды, состоящей из множества слабосвязанных разнородных вычислительных ресурсов. Рассматривается подход к организации системы метамониторинга, расширенной функциями управления. Подход основан на применении веб-технологий, мультиагентных технологий, методов создания экспертных систем, методов децентрализованной обработки и распределенного хранения данных.

1. Введение

Рассматриваемые в статье распределенные вычислительные среды (РВС) предназначены для организации ресурсоемких вычислительных экспериментов и включают высокопроизводительные системы различной архитектуры и конфигурации: высокопроизводительные серверы, включающие модули ускорения вычислений на базе графических ускорителей, сопроцессоров и ПЛИС, высокопроизводительные вычислительные кластеры (суперкомпьютеры), функционирующие под управлением различных систем управления прохождением задач (PBS/Torque, Cleo, СУППЗ), гибридные вычислительные системы, построенные на основе процессоров различной архитектуры и другие.

При построении таких РВС приходится решать ряд задач, связанных как с обеспечением слаженной работы всех вычислительных компонентов, так и с оснащением таких систем средствами управления и мониторинга. В задачи средств управления РВС входит распределение нагрузки на вычислительные узлы, выполнение непараллельных и параллельных команд с последующей передачей результата пользователю или оператору, выполнение загрузки и остановки работы вычислительных узлов и ряд других. Средства мониторинга должны обеспечивать оператора РВС структурированной и унифицированной информацией о состоянии любого вычислительного узла (использование вычислительного времени процессора, оперативной памяти, системы ввода-вывода, коммуникационной среды), о текущем состоянии оборудования, значениях датчиков (температура процессоров и материнских плат, температура в помещении), информации о работе устройств инженерной инфраструктуры и ряд других. С ростом числа вычислительных узлов в составе РВС возрастает и объём данных, которые приходится анализировать оператору, растёт вероятность ошибки вследствие человеческого фактора. Именно благодаря данным, предоставляемым системой мониторинга, их актуальности, наглядности и информативности является возможным своевременно реагировать на все негативные изменения в работе системы, выявлять причины неисправностей и сбоев.

Известные на сегодняшний день средства мониторинга имеют ряд недостатков, которые существенно затрудняют сбор и анализ данных для сложных вычислительных систем. В частности, в системе Ganglia отсутствует возможность использования механизма обработки текущих значений и уведомлений о событиях. Nagios и Zabbix лишены этого недостатка, однако изначально создавались для мониторинга сетевого оборудования и зачастую малоэффективны для мониторинга РВС. Отдельно следует выделить систему мониторинга LAPTA [1], предназначенную для всестороннего анализа динамических характеристик параллельных программ, выполняемых на суперкомпьютерах. Ориентация на сбор информации относительно ресурсов, используемых конкретной задачей, интеграция с очередями задач, предоставление агрегированной информации по задаче пользователя и ряд других функциональных возможностей, заявленных разработчиками, видятся нам наиболее верным развитием систем мониторинга для РВС и, в частности, вычислительных кластеров. Однако зависимость компонентов системы мо-

ниторинга LAPTA от ряда нестандартных интерпретируемых языков программирования и систем хранения данных значительно усложняет ее установку и настройку. Кроме того, реализация агента системы мониторинга на основе интерпретируемого языка программирования требует больше системных ресурсов узла РВС, что может приводить к замедлению работы счетных программ, выполняемых на этих узлах.

В данной работе предлагается подход к созданию инструментального комплекса метамониторинга РВС, отличающийся от известных уникальным набором свойств: автоматическим контролем программно-аппаратных ресурсов с использованием мультиагентных технологий, децентрализованной схемой хранения данных метамониторинга, применением экспертных систем для принятия решений.

Исходные требования к разрабатываемому инструментальному комплексу метамониторинга РВС [2] предполагают, что он должен:

- базироваться на принципах организации мультиагентных систем;
- обеспечивать высокую степень автономности компонентов системы;
- иметь многоуровневую иерархическую структуру, гарантирующую быстрое и надежное взаимодействие различных уровней, прямую и обратную связь между ними;
- обеспечивать возможность интеграции с наиболее популярными системами мониторинга высокопроизводительных ресурсов (Ganglia, Nagios и др.);
- предоставлять широкий набор функций сбора данных о состоянии программно-аппаратных компонентов вычислительных узлов, предоставлять средства разработки модулей сбора данных на различных языках системного программирования (C, Perl, Bash и др.);
- предоставлять средства сбора и анализа данных для оборудования вспомогательной (инженерной) инфраструктуры вычислительных кластеров;
- включать средства унификации данных, получаемых из различных источников;
- использовать технологию децентрализованного хранения данных, обеспечивающую снижение нагрузки на центральный узел системы метамониторинга;
- предоставлять средства автоматизированного экспертного анализа данных и генерации управляющий воздействий;
- включать средства визуализации, наглядно отображающие состояние ресурсов, задействованных в ходе решения задач;
- предоставлять прикладные программные интерфейсы на основе открытых стандартов для интеграции с другими программными комплексами.

2. Модель мультиагентной системы метамониторинга

Модель мультиагентной системы метамониторинга программно-аппаратных ресурсов (узлов) РВС можно представить в виде следующей структуры:

$$D = < M, N, T, A, R, S >,$$

где:

- M – множество измеряемых метрик (характеристик узлов);
- N – множество узлов РВС;
- T – множество технических средств коммуникации узлов РВС;
- A – множество агентов системы метамониторинга;
- R – множество правил вывода экспертной подсистемы;
- S – множество управляющих воздействий.

Связи между элементами множеств заданы отношениями:

- $NM \subseteq N \times M$ – отношение, определяющее множество метрик, измеряемых на узлах РВС;
- $NT \subseteq N \times T$ – отношение, определяющее топологию сети узлов РВС;
- $NA \subseteq N \times A$ – отношение, определяющее иерархическую структуру мультиагентной среды;
- $RM \subseteq R \times M$ – отношение, определяющее множество правил вывода, используемых экспертной подсистемой;
- $SM \subseteq S \times M$ – отношение, определяющее множество управляющих воздействий, генерируемых экспертной подсистемой по результатам анализа значений метрик.

Задача метамониторинга сформулирована следующим образом. Пусть $n_i \subset N$ – отдельный узел РВС. Для i-го узла измеряются m_i метрик ($m_i \subset M$). $B_i(t) = \{b_{ik}(t)\}$ – множество значений метрик для i-го узла в момент времени t, где $k = 1, 2, \ldots, m_i$.

Устойчивым состоянием метрик узла на отрезке времени $[t_0,t_1]$ считается такое, для которого выполняется неравенство:

$$| b_{ik}(t) - b_{ik}(t_0) | \leq c_{ik}, \forall t \in [t_0, t_1], \forall k \in [1, m_i],$$

где c_{ik} – положительная константа, характеризующая допустимый разброс значений для каждой метрики.

Значимым событием является переход узла из устойчивого состояния $B_i(t_0)$ в состояние $B_i(t_1)$, при котором происходит существенное изменение значения какой-либо метрики из множества m_i.

При возникновении значимого события на отрезке времени $[t_0, t_1]$ формируется перечень фактов для экспертной подсистемы, и на основе правил вывода R_i, определенных для узла i, машиной вывода генерируются управляющие воздействия S_i.

Под управляющим воздействием S_i понимается инструкция, предназначенная для реагирования на значимое событие. Среди таких управляющих воздействий могут быть:

- уведомление оператора системы метамониторинга;
- уведомление пользователя, задача которого выполняется в узле n_i;
- снятие счетных задач пользователя, выполняющихся в узле n_i;
- выключение узла n_i (к примеру, при длительном простое или выявлении критических значений заданных метрик);
- включение узла n_i (к примеру, при появлении задач в очереди СУПЗ).

Сгенерированное экспертной подсистемой управляющее воздействие передается управляющей подсистеме, где оно транслируется в системные команды и исполняется.

Далее состояние узла (B_i) передается агентам верхних уровней для анализа и генерации управляющих воздействий в соответствии с правилами, определенными на соответствующих уровнях.

3. Архитектура мультиагентной системы метамониторинга

Архитектура мультиагентной системы метамониторинга РВС включает следующие основные компоненты (**Рис. 1**):

- средства доступа пользователей, позволяющие взаимодействовать с системой метамониторинга как в пакетном, так и в интерактивном режимах;
- подсистемы уровня доступа, осуществляющие контроль прав доступа к запрашиваемым данным и реализующие серверную часть графического интерфейса пользователя;
- агент верхнего уровня, функционирующий в центральном узле РВС и выполняющий основную задачу по управлению системой метамониторинга;
- агенты промежуточного уровня, функционирующие на промежуточных узлах и решающие задачу снижения нагрузки на агенты верхних уровней;
- агенты нижнего уровня, функционирующие на исполнительных узлах РВС и осуществляющие сбор и первичную обработку данных о состоянии узла;
- подсистема децентрализованного хранения данных, предоставляющая функции для работы с данными для агентов разных уровней.

Иерархическая структура представляемой системы метамониторинга достигается путем организации многоуровневой мультиагентной среды, возглавляемой агентом верхнего уровня, которому подчинены агенты промежуточного и нижнего уровня. При этом агенты промежуточного уровня могут осуществлять управление как группой агентов нижнего уровня, так и группой агентов дополнительных промежуточных уровней, которых в системе может быть несколько. Такая иерархия позволяет обеспечить высокую масштабируемость системы метамониторинга для РВС, включающей большое число узлов.

Каждый агент мультиагентной системы метамониторинга реализован в виде программы, функционирующей в фоновом режиме, и обладает свойствами, характерными для автономных программных агентов [3], а именно, реактивностью, автономностью, целенаправленностью и коммуникативностью.

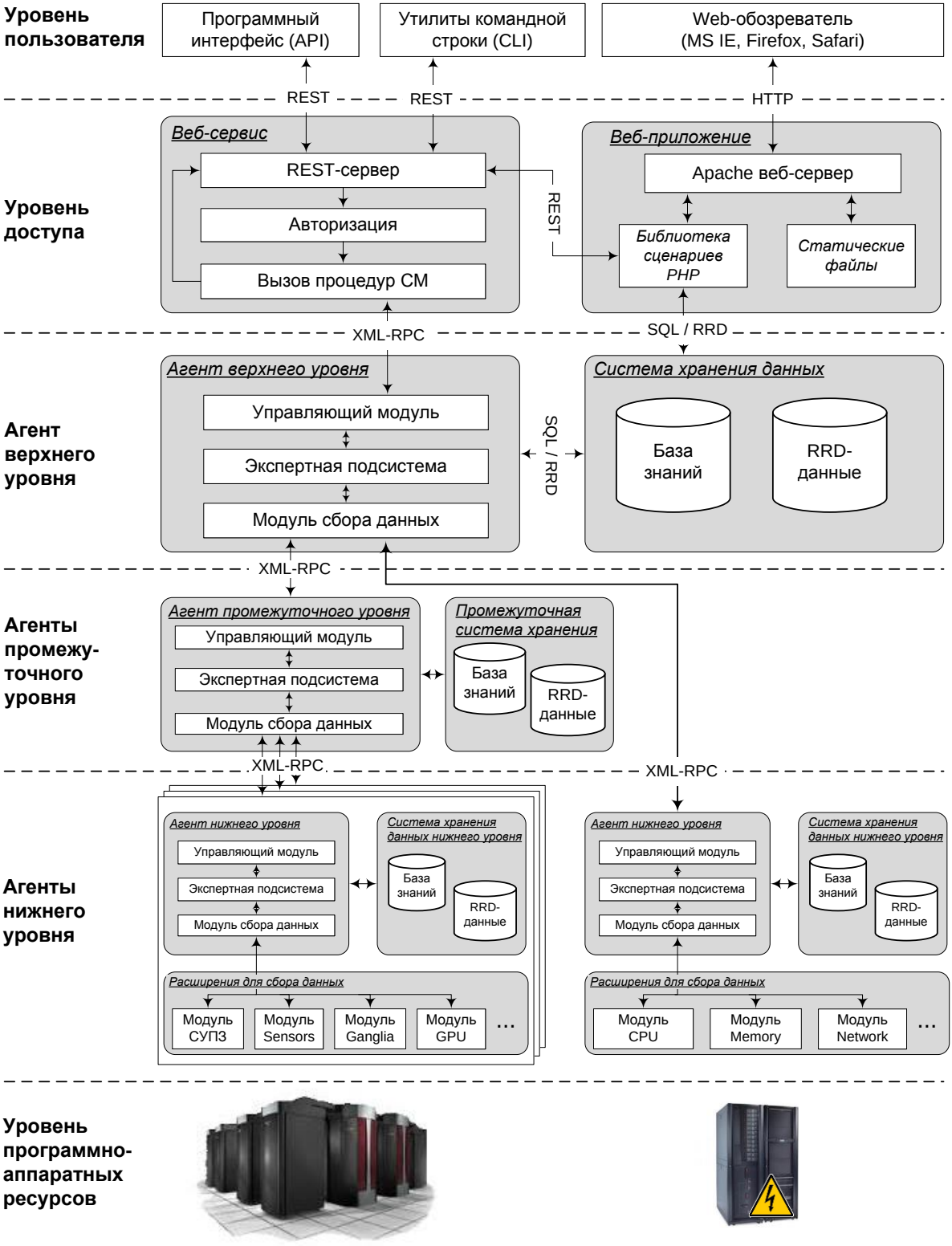

Рис. 1. Архитектура системы метамониторинга распределенной вычислительной среды

В составе агента системы метамониторинга реализованы следующие подсистемы:

- подсистема сбора данных, осуществляющая считывание показаний программных и аппаратных сенсоров различных устройств, прием данных от локальных систем мониторинга и агентов разных уровней, унификацию полученных данных и их трансляцию в промежуточный формат подсистемы взаимодействия с СУБД;
- подсистема взаимодействия с СУБД, выполняющая функции агрегации и контроля целостности данных;
- экспертная подсистема, выполняющая функции анализа данных, полученных за определенный интервал времени, и генерации управляющих воздействий на основе проведенного анализа;
- управляющая подсистема, выполняющая функции исполнения управляющих воздействий и взаимодействия с агентами верхних уровней.

Агент нижнего уровня устанавливается непосредственно в узле РВС и выполняет функции сбора данных о состоянии узла, анализа этих данных, генерации и исполнения управляющий воздействий, а также функции взаимодействия с другими агентами. Возможность анализа данных и принятия необходимых решений на стороне узла является ключевой особенностью реализуемого подхода. Как правило, клиенты систем мониторинга выполняют лишь функции сбора и периодической отправки данных в центральный узел, где производится обработка и анализ этих данных. При такой реализации существенно снижается количество операций по анализу данных на стороне узла, однако создается дополнительная нагрузка на стек сетевых протоколов, обработка которого (формирование, разбор и контроль сетевых пакетов) также требует процессорного времени. В нашем подходе анализ данных на стороне узла позволяет производить пересылку данных на центральный узел только в случае необходимости (либо по специальному запросу агентов верхних уровней). Установлено, что первичный анализ данных на стороне узла требует значительно меньше ресурсов, чем затраты на формирование сетевых пакетов и проверки их целостности. Сравнительный анализ работы агентов системы мониторинга Ganglia (gmond) и агентов разрабатываемой системы показал, что при одинаковой периодичности опроса датчиков (каждые 30 сек.), последний потребляет на 37% процессорного времени меньше. Кроме того, базовый набор сенсоров разрабатываемого агента значительно шире набора сенсоров агента системы мониторинга Ganglia. Таким образом, помимо снижения нагрузки на сеть и центральный узел системы метамониторинга удается снизить негативное влияние агента на производительность решения задач на узлах РВС.

Сбор информации о состоянии узла и загрузке отдельных его компонентов (процессора, памяти, сетевых интерфейсов и др.) реализован с использованием функций библиотеки SIGAR [4]. Реализация сторонних датчиков выполнена на специализированном языке, являющимся подмножеством языка ECMA Script [5], поддерживающим вызовы внешних команд, обработку выходного потока, регулярные выражения и ряд других механизмов, позволяющих получать данные о состоянии нестандартных программно-аппаратных устройств.

Агент промежуточного уровня устанавливается на промежуточных узлах РВС и выполняет функции сбора данных от группы агентов нижних уровней, агрегации и анализа этих данных, генерации и исполнения необходимых управляющих воздействий (в соответствии с правилами вывода, определенными для агента данного уровня), а также в случае необходимости осуществляет отправку данных агентам верхних уровней. Основное назначение агента промежуточного уровня – снижение нагрузки на агенты верхних уровней.

Агент верхнего уровня устанавливается в центральном узле РВС и выполняет функции ядра системы, владеет полной информацией о состоянии улов РВС и предоставляет средства доступа к данным системы метамониторинга. Агент верхнего уровня может взаимодействовать с агентами промежуточного уровня, с агентами нижнего уровня, а также напрямую получать данные от локальных систем мониторинга и отдельных сенсоров. Собираемая информация накапливается в центральной базе данных, унифицируется, агрегируется и предоставляется для доступа экспертной подсистеме и подсистемам уровня доступа. Экспертная подсистема агента верхнего уровня анализирует общее состояние РВС, генерирует управляющие воздействия, которые далее исполняются управляющей подсистемой. В базе знаний экспертной системы содержится информация о том, какие управляющие воздействия должны приниматься в автоматическом режиме, а какие при участии оператора РВС.

Экспертная подсистема. Для реализации функций экспертной подсистемы использована популярная среда CLIPS [6], основанная на продукционной модели знаний и предоставляющая программный интерфейс для работы с основными элементами экспертной подсистемы: базой знаний, содержащей списки фактов, списки объектов и правила вывода; механизмом вывода, выполняющим логический вывод на основе сформированной базы знаний.

Для анализа значений метрик выбрана классификация по степени критичности, в соответствии с которой для каждой метрики определено четыре состояния: ошибка, критическое, предупреждение, удовлетворительное.

Значения метрик обрабатываются набором правил, которые оценивают эти значения в соответствии с введенными граничными параметрами и присваивают каждой метрике флаг состояния. Если в списке фактов появляется хотя бы одно состояние "ошибка" или "критическое", то для избранных метрик проверяется история значений за определенный период, чтобы исключить явления, несущие случайный характер и неспособные в какой-либо мере повлиять на работу системы. Если информация о состоянии ошибки или критическом состоянии подтверждается, то генерируются управляющие воздействия, среди которых могут быть следующие: информирование агента верхнего уровня; создание и отправка сообщений оператору системы; действия, направленные на автоматическое решение возникшей проблемы.

Подсистема накопления и обработки данных основана на принципах функционирования циклических баз данных (Round Robin Database) – баз данных, объем которых не меняется со временем. Фиксированный размер таких баз достигается благодаря заранее определенному количеству записей, используемых циклически для сохранения данных.

Несмотря на неоспоримые преимущества циклических баз данных, наиболее известные реализации (MRTG, RRDtools и др.) имеют ряд недостатков, связанных, прежде всего, с низкой производительностью операций чтения/записи данных.

Проведенные эксперименты по созданию циклической базы данных на основе легковесной встраиваемой реляционной базы данных SQLite выявили еще более худшую производительность таких баз данных в сравнении с RRDtools и MRTG. В результате было принято решение о создании собственной реализации циклической базы данных, использующей для хранения структурированной информации специализированный текстовый формат, основанный на расширяемом метаязыке XML. Были реализованы механизмы чтения/записи данных, агрегации данных за определенный интервал времени, вытеснения устаревших данных, выборки данных по определенным критериям, а также механизмы кеширования данных. Собственная реализация циклической базы данных показала высокую эффективность в сравнении с универсальными системами организации циклических баз данных (RRDtools, MRTG).

Для хранения правил вывода экспертной подсистемы, списка фактов, сгенерированных управляющих воздействий, шаблонов для трансляции управляющих воздействий в системные команды и других знаний используется легковесная встраиваемая реляционная база данных SQLite. Для выполнения перечисленных функций производительность SQLite является приемлемой, а возможность встраивания средств управления базой данных в состав программного агента позволяет сохранить малый размер программы и обеспечить минимальную зависимость от стороннего программного обеспечения, установленного в узле.

Графический пользовательский интерфейс является одним из наиболее важных компонентов любой системы, предполагающей взаимодействие с пользователем. Информативность и наглядность представления данных, актуальность отображаемой информации, возможность формирования комплексных отчетов по выбранным параметрам и ряд других возможностей делают систему полезной для различных категорий пользователей.

В качестве одного из существенных отличий предлагаемого подхода к построению интерфейса системы метамониторинга является реализация средств генерации графиков на клиентской стороне веб-приложения (в веб-браузере пользователя). Такой подход, во-первых, позволяет снизить нагрузку на центральный узел системы метамониторинга за счет переноса части нагрузки на компьютер пользователя. Во-вторых, при таком подходе появляется возможность отображать графики в режиме реального времени без периодической перезагрузки веб-страниц. В-третьих, в значительной мере снижается объем передаваемого трафика между клиентской и серверной сторонами веб-приложения.

В качестве языков разработки пользовательского веб-интерфейса были выбраны: язык про-

граммирования PHP для реализации серверной части веб-приложения и совокупность веб-технологий, языков программирования и библиотек (таких как HTML5, JavaScript, Ajax, ExtJS, jQuery) для реализации клиентской части веб-приложения. Пример отображения графиков в режиме реального времени с использованием графических средств библиотеки ExtJS приведен на **Рис. 2**.

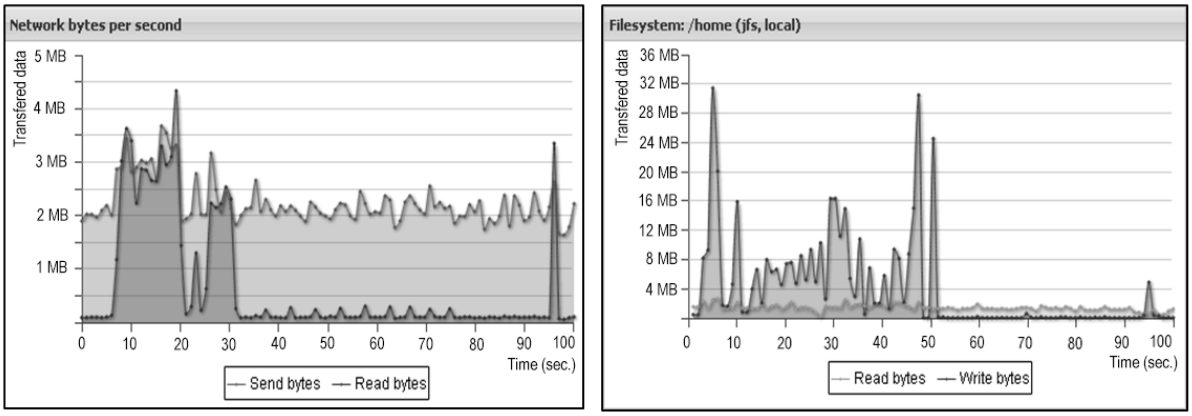

Рис. 2. Графики текущей загрузки сетевого интерфейса (слева) и файловой системы (справа)

На **Рис. 3** приведена схема размещения компонентов систем холодоснабжения и электроснабжения [7] вычислительного кластера «Академик В.М. Матросов» в машинном зале (слева) и на внешней площадке (справа) с отображением значений ключевых метрик. На **Рис. 4** в табличной форме представлен расширенный список значений метрик.

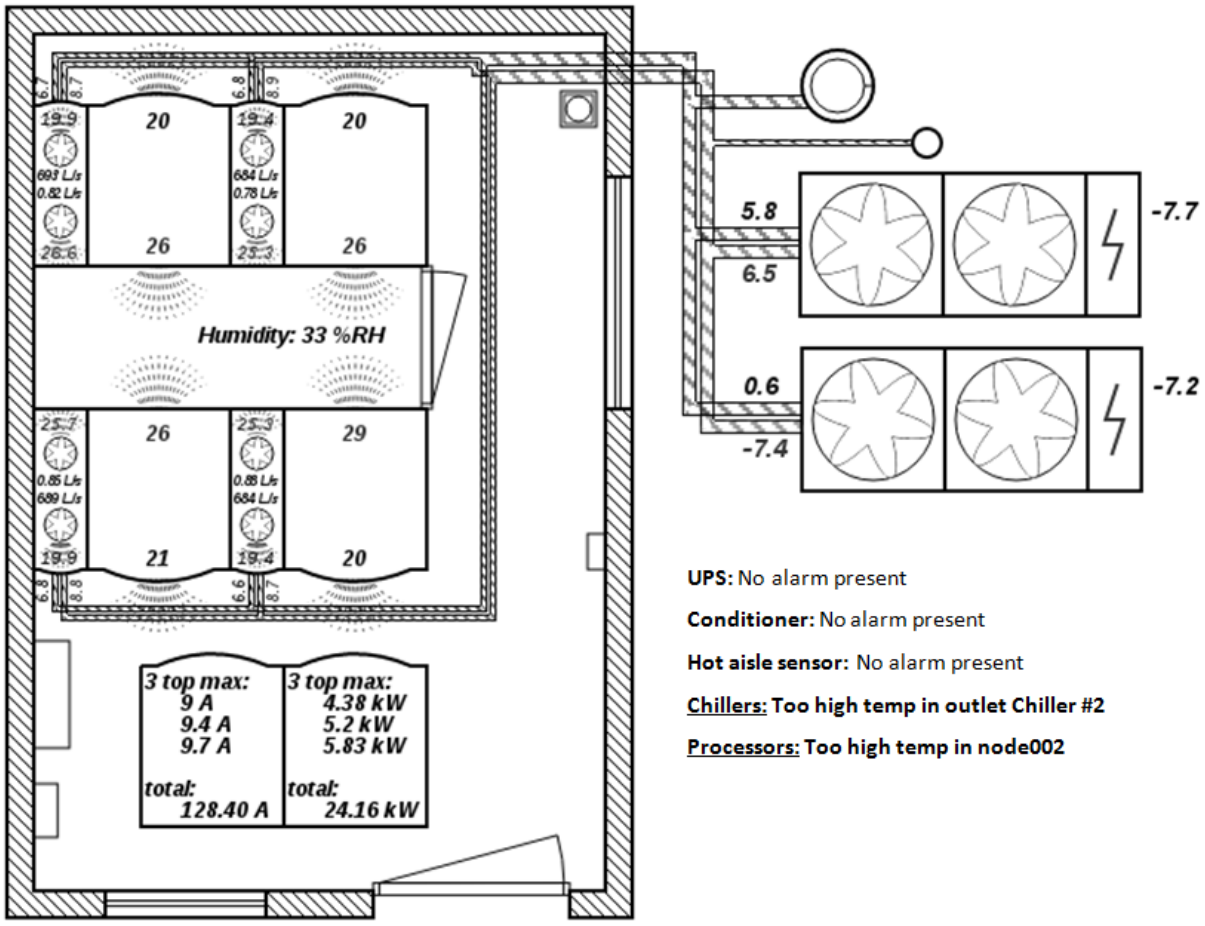

Рис 3. Схема размещения компонентов систем холодоснабжения и электроснабжения с указанием значений ключевых метрик

Chillers

Параметры	# 1	# 2
Статус чиллера	**Актив**	Резерв
Темп. вход контура	6.5 °C	3.1 °C
Темп. испарителя	2.9 °C	4.9 °C
Темп. выход контура	3.0 °C	4.0 °C
Темп. окр. воздуха	-14.5 °C	-18.0 °C
Заданные значения	6.5 °C	6.5 °C
Статус насоса	2	0
Естеств. охлаждение	0 %	0 %
Компрессоры	0 %	0 %
Отопление	0 %	0 %

Hot aisle

Humidity	29 %
R2-Front	21 °C
R1-Front	20 °C
R2-Rear	31 °C
R1-Rear	34 °C
R3-Front	20 °C
R4-Front	20 °C
R3-Rear	32 °C
R4-Rear	30 °C

CPU Temperature

Node	Proc	Temp
node070	CPU1	64.00
node063	CPU1	64.00
node061	CPU1	63.00
node070	CPU0	63.00
node069	CPU0	63.00
node061	CPU0	63.00
node069	CPU1	62.00
node068	CPU1	62.00
node065	CPU0	62.00

CPU Frequency

node001	2300.000
node007	2300.000
node064	2300.000
node060	2300.000

CPU Memory

node051	**56345.594**
node064	64420.707
node076	64420.707
node002	64425.570

Conditioners

Статус работы	On	On	On	On
Выходная мощность охлаждения	12.0 kW	10.7 kW	13.3 kW	9.4 kW
Необход. мощность охлаждения	12.0 kW	10.7 kW	13.3 kW	9.4 kW
Температура на входе	**21.3 °C**	**20.9 °C**	**21.3 °C**	**22.2 °C**
Темп. возвращаемого воздуха	20.1 °C	19.9 °C	20.0 °C	20.1 °C
Темп. приточного воздуха	**30.3 °C**	**30.9 °C**	**30.3 °C**	**29.8 °C**
Поток воздуха	**829 L/s**	**834 L/s**	**834 L/s**	**834 L/s**
Скорость вентиляторов	**53.4** %	**53.9** %	**53.9** %	**53.9** %
Состояние на входе	Open	Open	Open	Open
Состояние на выходе	Closed	Closed	Closed	Closed
Активный источник питания	A	A	A	A
Перепад давления на фильтре	0 Pa	5 Pa	17 Pa	15 Pa
Положение клапана в жидкости	**54** %	**54** %	**56** %	**57** %
Поток жидкости	**0.85 L/s**	**0.81 L/s**	**0.88 L/s**	**0.76 L/s**
Температура жидкости на входе	**6.2 °C**	**6.2 °C**	**5.9 °C**	**6.7 °C**
Температура жидкости на выходе	9.9 °C	9.6 °C	9.8 °C	9.9 °C
Общие установ. значения темп.	**22.2 °C**	**22.2 °C**	**22.2 °C**	**22.2 °C**
Общие установленные значения	20.0 °C	20.0 °C	20.0 °C	20.0 °C

UPS

#	Rating	Current	Power
1	16A	1.1 A	0.21 kW
	16A	0.0 A	0.00 kW
	16A	1.2 A	0.22 kW
2	16A	1.1 A	0.19 kW
	16A	0.0 A	0.00 kW
	16A	1.1 A	0.21 kW
3	32A	15.8 A	
	32A	9.4 A	8.38 kW
	32A	13.7 A	
4	32A	16.4 A	
	32A	18.9 A	10.47 kW
	32A	11.9 A	
5	**32A**	**15.8 A**	
	32A	**16.6 A**	**10.65 kW**
	32A	**15.7 A**	
6	**32A**	**15.3 A**	
	32A	**9.8 A**	**8.39 kW**
	32A	**13.3 A**	

Рис 4. Расширенный список значений метрик

4. Вычислительный эксперимент

Разработанные методы и инструментальные средства метамониторинга РВС были успешно апробированы в суперкомпьютерном центре ИДСТУ СО РАН [8] при решении ресурсоемких задач в области биоинформатики, квантовой химии, логического поиска и других.

Конфигурация экспериментальной РВС включала следующие программно-аппаратные ресурсы:

- 2 вычислительных узла с графическими ускорителями Nvidia Tesla C1060 с общим числом ядер 1920;
- 10 вычислительных узлов с процессорами AMD Opteron 6276 с общим числом ядер 320;
- 20 вычислительных узлов с процессорами Intel Xeon 5345 с общим числом ядер 160.

В процессе тестирования системы метамониторинга в РВС был выявлен перечень программно-аппаратных ресурсов, находившихся в критическом состоянии и в состоянии ошибки:

- (warning node node-4.blackford.icc.ru loadavg 43);
- (critical node node-13.blackford.icc.ru cpu-sys-p 77);
- (critical node node011.matrosov.icc.ru filesystem /store wtime 583816);
- (critical node node020.matrosov.icc.ru filesystem /store wtime 469986);
- (error node node-4.blackford.icc.ru available down).

– (warning node node-15.blackford.icc.ru memory-used-ten 81);
– (critical node node2.tesla.icc.ru memory-used-ten 92);
– (error node node-7.blackford.icc.ru filesystem /store du-free 0).

Анализ сведений о состоянии программно-аппаратных ресурсов РВС, собранных системой метамониторинга, позволил выявить неэффективную работу пользовательских приложений, произвести оптимизацию загрузки вычислительных ресурсов и повысить показатели надежности РВС.

Так, к примеру, при аннотации генома Synedra acus программным пакетом MAKER [9] было выявлено преобладание операций чтения записи в сетевую директорию по отношению к вычислительным операциям, выполняемым на процессорных ядрах. Установка параметров пакета, указывающих размещение директорий для записи результатов вычислений, в значения локальных директорий узлов (в частности, /tmp) позволило повысить эффективность работы пакета более чем на 30%.

5. Заключение

Оригинальность и новизна предложенного подхода заключается в первую очередь в создании универсальных программных агентов, способных собирать данные о состоянии узла, анализировать их и принимать необходимые решения непосредственно самим агентом. Мультиагентный подход позволит, во-первых, снизить общую нагрузку, создаваемую компонентами системы мониторинга, во-вторых, повысить надежность РВС за счет децентрализованной генерации и исполнения управляющих воздействий. Универсальность архитектуры создаваемого программного агента позволит использовать его на различных уровнях иерархии системы мониторинга, что в свою очередь позволит обеспечить высокую масштабируемость при использовании в РВС, включающих большое количество узлов.

Литература

1. Адинец А.В., Брызгалов П.А., Воеводин Вад.В. Мониторинг, анализ и визуализация потока заданий на кластерной системе // Материалы XI Всерос. конф. "Высокопроизводительные параллельные вычисления на кластерных системах". – Нижний Новгород, 2011. – С. 10-14.
2. Сидоров И.А., Скоров В.В. Один из подходов к реализации средств мониторинга высокопроизводительных вычислительных систем // Информационные и математические технологии в науке и управлении: Тр. XVIII Байкальской Всерос. конф. – Ч. III. – Иркутск: Изд-во ИСЭМ СО РАН, 2013. – С. 177-184.
3. Wooldridge M., Jennings N. Intelligent Agents: Theory and Practice // The Knowledge Engineering Review. – 1995. – Vol. 10, № 2. – P. 115-152.
4. System Information Gatherer and Reporter API. URL: http://www.hyperic.com/products/sigar (дата обращения: 29.11.2013).
5. Standard ECMA-262: ECMAScript Language Specification. URL: http://www.ecma-international.org/publications/standards/Ecma-262.htm (дата обращения: 29.11.2013)
6. Частиков А.П. Разработка экспертных систем. Среда CLIPS / А. П. Частиков, Д. Л. Белов, Т. А. Гаврилова. – Санкт-Петербург: БХВ-Петербург, 2003. – 393 с.
7. Скоров В.В., Сидоров И.А., Новопашин А.П. Редактор графического представления объектов инженерной инфраструктуры вычислительного центра // Материалы конф. “Ляпуновские чтения”. – Иркутск: Изд-во ИДСТУ СО РАН, 2013. – С. 53.
8. Иркутский суперкомпьютерный центр СО РАН. URL: http://hpc.icc.ru (дата обращения: 29.11.2013)
9. MAKER – genome annotation pipeline. URL: http://gmod.org/wiki/MAKER (дата обращения: 29.11.2013)