

РАЗРАБОТКА ПРИНЦИПОВ ПОСТРОЕНИЯ И РЕАЛИЗАЦИЯ ПРОТОТИПА СИСТЕМЫ ОБЕСПЕЧЕНИЯ ОПЕРАТИВНОГО КОНТРОЛЯ И ЭФФЕКТИВНОЙ АВТОНОМНОЙ РАБОТЫ СУПЕРКОМПЬЮТЕРНЫХ КОМПЛЕКСОВ

А. С. Антонов¹, Вад. В. Воеводин², Вл. В. Воеводин³, С. А. Жуматий⁴,
Д. А. Никитенко⁵, С. И. Соболев⁶, К. С. Стефанов⁷, П. А. Швец⁸

¹asa@parallel.ru, ²vadim@parallel.ru, ³voevodin@parallel.ru, ⁴serg@parallel.ru, ⁵dan@parallel.ru,
⁶sergeys@parallel.ru, ⁷cstef@parallel.ru, ⁸shvets.pavel.srcc@gmail.com

Научно-исследовательский вычислительный центр
Московского государственного университета имени М. В. Ломоносова (НИВЦ МГУ)

Поступила в редакцию 23.12.2013

Аннотация. Современный суперкомпьютерный комплекс – это сложная, дорогостоящая и крайне энергонасыщенная система. Каждый из компонентов комплекса потенциально ненадежен и может выйти из строя практически в любой момент. Последствия могут быть различными, от завершения отдельных заданий до выхода из строя всего комплекса, поэтому необходим оперативный контроль, позволяющий гарантировать сохранность компонентов и эффективную автономную работу суперкомпьютера в целом. В НИВЦ МГУ разрабатывается программный комплекс для оперативного контроля и автоматического реагирования на аварийные ситуации в работе суперкомпьютерных систем. В основу комплекса положена модель суперкомпьютерной системы, представленная в виде мультиграфа. Такой подход позволяет обеспечить полноту определения аварийных ситуаций, а также вести учет накопленного опыта эксплуатации больших и сверхбольших вычислительных систем. Работа рекомендована Программным комитетом Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: все грани параллелизма».

Ключевые слова: суперкомпьютер; надежность суперкомпьютера; отказоустойчивость суперкомпьютера; мониторинг суперкомпьютера; аварии суперкомпьютера; отказы суперкомпьютера; автономная работа суперкомпьютера; модель функционирования суперкомпьютера.

ВВЕДЕНИЕ

Устойчивый рост производительности суперкомпьютерных комплексов, как уже существующих, так и только проектируемых, сопровождается неизбежным увеличением числа и сложности их компонентов. Это относится как к оборудованию, включающему в себя вычислительные и коммуникационные модули, подсистемы охлаждения, энергообеспечения и т. д., так и к программной инфраструктуре, на которую ложится существенная задача по поддержке эффективной согласованной работы всех компонентов комплекса. Чем сложнее система, чем

больше компонентов в ней, тем сложнее решается задача обеспечения ее надежности и корректного и эффективного функционирования. Эта проблема в настоящее время актуальна для всех суперкомпьютерных комплексов, обслуживающих сотни и тысячи пользователей, которые должны быть обеспечены доступом к высокопроизводительным ресурсам в режиме 24/7.

Актуальность проблемы определяется как существенной стоимостью суперкомпьютерных систем, так и их высокой востребованностью вычислительным сообществом. Стоимость суперкомпьютеров значительна, спрос на суперкомпьютерные ресурсы велик, значит, необходимо увеличивать их отдачу, минимизируя простои оборудования, отдавая в каждый момент в работу все то, что можно отдать пользователям для расчетов. Если какая-то часть суперкомпью-

ютера неисправна, ее нужно локализовать и вывести из конфигурации, оставив в работе максимально возможную часть нормально функционирующего оборудования для продолжения обслуживания потока задач.

Вместе с этим современные суперкомпьютерные комплексы обладают двумя критически важными свойствами, определяющими сложность их построения и последующего сопровождения: чрезвычайно большое число компонентов и высокий уровень энергопотребления.

Первое свойство – большое число компонентов – приводит к тому, что в суперкомпьютерах постоянно какие-то компоненты выходят из строя. Неисправные компоненты будут всегда, поэтому их негативное влияние на работу всего суперкомпьютерного комплекса должно быть оперативно минимизировано, а машина должна продолжать эффективно работать. Неисправные узлы нужно уметь оперативно обнаруживать и выводить из счетного поля для обеспечения эффективной работы суперкомпьютерного комплекса в целом. Важно, что подобного рода работа должна выполняться постоянно и автоматически.

Второе свойство современных суперкомпьютерных комплексов – высокий уровень энергопотребления в совокупности с высокой плотностью размещения компонентов – приводит к необходимости построения серьезных систем охлаждения и постоянного оперативного отвода большого объема тепла. Сбой в системах охлаждения может привести не только к снижению эффективности работы суперкомпьютерного комплекса или к выходу из строя части дорогостоящего оборудования, но и к возникновению опасных аварийных ситуаций. Например, технология «горячих коридоров», используемая во многих суперкомпьютерных центрах, требует постоянного внимания к своей работе: в случае отключения систем охлаждения температура внутри горячего коридора будет повышаться примерно на 20 градусов каждую минуту, что уже через несколько минут безвозвратно погубит все оборудование. В такой ситуации отсутствие контроля над функционированием суперкомпьютерного комплекса даже в течение 5 минут может иметь самые серьезные и необратимые последствия. Нужна постоянно работающая система, которая должна отслеживать большое число аппаратных датчиков (температуры, влажности, утечки охлаждающей жидкости и т.п.), самостоятельно принимающая решение относительно необходимости отключения отдельных компонентов, подсистем, а иногда и всего суперкомпьютерного комплекса.

Очень важный момент – полнота охвата такой системой всего спектра возможных аварийных ситуаций. В ней должен найти отражение весь предшествующий опыт сопровождения суперкомпьютерных комплексов. Это позволит не совершать два раза одни и те же ошибки, накапливая как собственный опыт, так и опыт других коллективов в сопровождении суперкомпьютерных систем. А опыт уже богат, и примеров тому много. Датчик протечки охлаждающей жидкости может молчать и не сигнализировать о проблеме не потому, что все в порядке, а потому, что он вышел из строя. Система оповещения по SMS может не присылать тревожных сигналов не потому, что все в порядке, а потому, что на счету просто закончились деньги. Аварийное отключение систем охлаждения было вовремя замечено автоматикой, но оказалось, что после последней перезагрузки головного узла суперкомпьютера служебный процесс, отвечающий за выключение оборудования, не стартовал... Подобных ситуаций очень много, все они потенциально опасны и требуют незамедлительной реакции.

Традиционные методы обеспечения надежности суперкомпьютерного комплекса заключаются в выборе, настройке и использовании одной из доступных систем мониторинга и визуализации состояния комплекса (Nagios [1], Ganglia [2], Zabbix [3], Zenoss [4] и др.). Мониторинг инфраструктурного оборудования может обеспечиваться программными решениями его производителей. Наконец, для учета специфических особенностей комплекса и определения наиболее опасных или характерных аварийных ситуаций создается набор сценариев («скриптов»), которые срабатывают либо периодически, либо по наступлению какого-либо события, и выполняют ряд специализированных проверок (доступна ли очередь заданий, все ли вычислительные узлы находятся в готовности и т.д.).

Однако такой подход ориентирован прежде всего на сбор информации о состоянии множества отдельных компонентов вычислительного комплекса, в основном относящихся к его вычислительным модулям, и их минимальную агрегацию и визуализацию. Логика по выявлению сбоев и реагированию на аварийные ситуации фактически отдается на откуп системным администраторам комплекса, которые и реализуют ее в виде набора дополнительных программных модулей. Подобные решения практически не являются переносимыми между различными вычислительными комплексами, они не поддерживают накопление истории возникновения критических ситуаций и ее анализ, формирова-

ние банков аварий и методов реагирования на них.

О ПРОЕКТЕ

В начале 2013 г. в НИВЦ МГУ им. М.В. Ломоносова стартовал проект по созданию принципов построения и реализации системы обеспечения оперативного контроля и эффективной автономной работы суперкомпьютерных комплексов. Разработанную систему планируется внедрить и использовать на имеющихся в МГУ суперкомпьютерах «Чебышев» (60 TFlops) и «Ломоносов» (1.7 PFlops) [5–8], а также на перспективных высокопроизводительных системах, планируемых к установке в МГУ. Основные задачи проектируемого программного комплекса сформулированы следующим образом:

- обеспечение максимальной сохранности оборудования вычислительного комплекса;
- обеспечение максимально полного использования оборудования;
- поддержка преемственности опыта сопровождения суперкомпьютерных систем.

В основе проекта лежит идея построения модели функционирования современного суперкомпьютерного комплекса, отражающей основные компоненты суперкомпьютеров и их взаимосвязь, ориентированной на повышение продуктивности работы суперкомпьютерных систем, на оперативное обнаружение и локализацию аварийных ситуаций. Наличие адекватной модели позволяет создать систему, гарантированно контролирующую все аспекты работы комплекса. Модель функционирования суперкомпьютерного комплекса позволит оценить степень влияния аварийной ситуации на смежные блоки и подсистемы суперкомпьютера, ми-

нимизировать объем автоматически отключаемого оборудования и поддерживать максимально возможную часть нормально функционирующего оборудования в работоспособном состоянии.

Исходя из многолетнего опыта коллектива разработчиков по проектированию и сопровождению суперкомпьютерных систем [9–10], к разрабатываемой системе были установлены следующие требования:

- Обнаружение всех потенциальных источников отказов комплекса. Помимо очевидных причин отказов и аварий, которые могут быть определены исходя из структуры модели, создается банк нетривиальных аварийных ситуаций, в котором аккумулируется реальный опыт поддержки больших вычислительных комплексов. Такой банк, в частности, упростит верификацию модели.
- Наличие средств реагирования на аварийные ситуации, включающие как их автоматическое устранение (отключение или перезагрузка оборудования, перезапуск программных компонентов), так и оповещение персонала.
- Гибкость и независимость от целевой архитектуры. Система должна быть настраиваемой на работу с суперкомпьютерным комплексом, построенным на основе наиболее распространенных программно-аппаратных архитектур.
- Масштабируемость. Система должна как поддерживать суперкомпьютеры текущего поколения тера- и петафлопсной производительности, так и быть готовой к работе с вычислительными комплексами следующего поколения, в которых число компонентов будет увеличено на порядки.

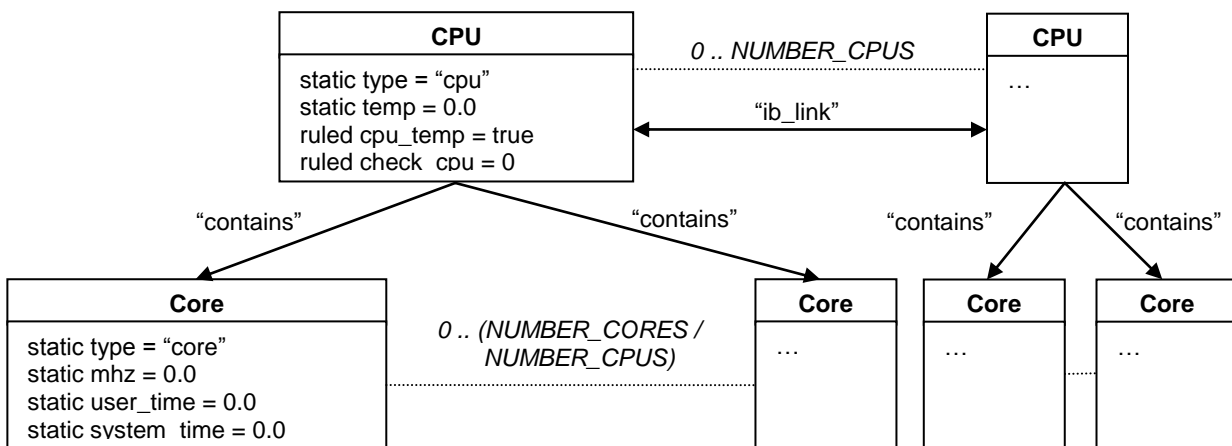


Рис. 1. Тестовая модель вычислительной системы в виде графа

```

1 graph = new CNeo4jGraph("dbs/graphit_10", CNeo4jGraph.Op.RECREATE);
2 graph.GetIndex().EnableObjectIndex("type");
3 IAutoRule cpu_temp = new CUpperThreshold("check_cpu_temp", "CPU", "temp", 50.0);
4 IAutoRule check_cpu = new CCheckBoolRules("cpu_errors", cpu_temp);
5
6 IAttributeN cpu_attributes = new IAttribute[] {
7     new CStaticAttribute("type", "cpu"),
8     new CStaticAttribute("temp", 0.0),
9     new CRuledAttribute(cpu_temp, true),
10    new CRuledAttribute(check_cpu, 0) };
11 IAttributeN core_attributes = new IAttribute[] {
12    new CStaticAttribute("type", "core"),
13    new CStaticAttribute("mhz", 0.0),
14    new CStaticAttribute("user_time", 0.0),
15    new CStaticAttribute("system_time", 0.0) };
16
17 IAutoList<IAutoObject> cpu = new CNodeFactory(graph, true, cpu_attributes)
18     .Create(NUMBER_CPUS);
19 IAutoList<IAutoObject> cores = new CNodeFactory(graph, true, core_attributes)
20     .Create(NUMBER_CORES);
21
22 CRelationshipConnectorFactory contains = new CRelationshipConnectorFactory
23     (graph, new CStaticAttribute("type", "contain"));
24 CRelationshipConnectorFactory ib_link = new CRelationshipConnectorFactory
25     (graph, new CStaticAttribute("type", "ib"));
26
27 contains.EveryToChunks(cpu, cores);
28 ib_link.AllToAll(cpu, cpu);

```

Рис. 2. Пример описания модели тестовой вычислительной системы на языке Java

- Наличие средств самодиагностики. На практике регулярно возникают ситуации, когда имеющиеся системы мониторинга и обеспечения отказоустойчивости не предупреждают об аварийных ситуациях из-за отказа датчиков, проблем с собственным функционированием и т. д. Очевидно, что механизмы самодиагностики должны быть заложены внутри разрабатываемой системы. При этом они также должны опираться на модель суперкомпьютерного комплекса, контролируя корректность работы всех доступных средств мониторинга и оповещения.

- Обеспечение преемственности в сопровождении суперкомпьютерных комплексов. Это требование подразумевает учет накопленного опыта, выраженного в нашем случае в виде связки модели комплекса и банка аварийных ситуаций, при установке на новые вычислительные системы.

- Наличие удобного интерфейса для работы с системой, в том числе средств визуализации отдельных компонентов, логических и физических блоков суперкомпьютерного комплекса.

МОДЕЛЬ СУПЕРКОМПЬЮТЕРНОГО КОМПЛЕКСА

В основу представления модели функционирования суперкомпьютерного комплекса положена структура мультиграфа.

Вершины мультиграфа, или объекты, соответствуют всем программным или аппаратным компонентам суперкомпьютера, которые необходимо контролировать (горячие коридоры, стойки, ИБП, процессоры, процессорные ядра и т. д.). Каждому объекту приписывается набор атрибутов – некоторых характеристик объекта, которые будет отслеживать создаваемая система. Все атрибуты разделяются на независимые и вычисляемые. Значения независимых атрибутов определяется только значениями датчиков – внешних источников информации о состоянии и поведении компонентов суперкомпьютера. В качестве источника данных будет выступать, в частности, установленная на суперкомпьютере система (или системы) мониторинга. Значения вычисляемых атрибутов формируются на основе других атрибутов после вызова определенных функций. К примеру, температура процессора вычислительного узла является независимым атрибутом, а свойство загруженности узла, определяемое на основе текущей загрузки всех ядер его процессора, будет вычисляемым атрибутом.

Ребра мультиграфа соответствуют физическим или логическим связям между соответствующими компонентами суперкомпьютера.

```

1 rule CUpperThreshold cpu_temp ("check_cpu_temp", "CPU", "temp", 50.0);
2 rule CCheckBoolRules check_cpu ("cpu_errors", cpu_temp);
3
4 group cpu_attributes {
5     static ("type", "cpu"),
6     static ("temp", 0.0),
7     ruled (cpu_temp, true),
8     ruled (check_cpu, 0) };
9
10 group core_attributes {
11     static ("type", "core"),
12     static ("mhz", 0.0),
13     static ("user_time", 0.0),
14     static ("system_time", 0.0) };
15
16 list[NUMBER_CPUS] cpu = object(cpu_attributes);
17 list[NUMBER_CORES] cores = object(core_attributes);
18
19 link_type contains;
20 link_type ib_link;
21
22 contains.EveryToChunks(cpu, cores);
23 ib_link.AllToAll(cpu, cpu);

```

Рис. 3. Пример описания тестовой модели вычислительной системы на специальном языке

Это могут быть связи типа «содержит» (шасси_1 содержит вычислительный_узел_1,..., вычислительный_узел_N), «питает» (ИБП_1 обеспечивает питанием стойку_1), «охлаждает» (кондиционер_1,..., кондиционер_N отводят тепло из горячего коридора_1), «Infiniband», «Ethernet» (узел_1,..., узел_N соединены с коммутатором_1). Допускается создание связей любого типа.

В качестве хранилища и представления графовой модели вычислительной системы выбрана open-source графовая СУБД Neo4j [11], реализованная на языке Java. Разработан API для построения модели вычислительного комплекса, содержащей основные типы объектов и связей между ними. С использованием этого API в программе на Java генерируется модель конкретной вычислительной системы.

Рассмотрим простую тестовую модель вычислительной системы, содержащую объекты типа «процессор» с вычисляемыми атрибутами «температура процессора» и «ошибки процессора» и объекты «процессорные ядра» с независимыми атрибутами «тактовая частота», «user_time» и «system_time», при этом между процессорами и заданным числом ядер установлена связь типа «содержит», а между процессорами — связь типа «Infiniband» (рис. 1). На рис. 2 приведено описание этой модели на языке Java с помощью разработанного API. Опишем по строкам процесс создания модели:

1: Создание пустой модели.

3–4: Объявление функций контроля температуры и проверки ошибок CPU. Пока эти функции не привязаны ни к одной вершине графа.

6–15: Объявление атрибутов, которые пока также не привязаны к вершинам графа.

17–20: Создание вершин графа, соответствующих процессорам и ядрам реальной системы. Также одновременно выполняется привязка атрибутов к данным вершинам.

22–25: Описание типов связей, которые будут использоваться в данной модели.

27: Создание связей типа «содержит» между процессорами и соответствующими ядрами.

28: Создание связей типа «сеть Infiniband» между всеми процессорами (топология «все-ко-всем»).

Описание модели вычислительного комплекса в виде программы на языке Java не всегда является удобным. Поэтому параллельно разрабатывается язык описания моделей. В качестве требований к разрабатываемому языку предъявляются простота и удобство описания, легкая читаемость и возможность генерации внешними автоматическими средствами. На рис. 3 приведен фрагмент программы на данном языке, создающий модель, аналогичную описанной выше. Описание модели на этом языке транслируется в Java-программу. Язык содержит конструкции для создания объектов и связей между ними.

```

1 public class CAESSTempRule extends CBaseRule
2 {
3     int lower, upper, num_out;
4
5     @Override
6     public EDependencyType GetDeps()
7     {
8         return EDependencyType.IN;
9     }
10
11    @Override
12    public Object Compute(IAutoObject object)
13        throws ExceptionModelFail
14    {
15        int count
16            = object.GetInNeighbors("type", "chill")
17              .GetAttr("ColdTemp").ge(upper).size()
18            + object.GetInNeighbors("type", "chill")
19              .GetAttr("ColdTemp").le(lower).size();
20
21        return count >= num_out;
22    }
23 }

```

Рис. 4. Пример функции вычисления атрибутов

Имеется возможность подгружать данные из внешних файлов в формате CSV. Например, если имеется CSV-файл, содержащий список имен и сетевых адресов узлов конкретного суперкомпьютера, то одной языковой конструкцией можно сопоставить данные из этого файла и значения соответствующих атрибутов объектов типа «узел» модели, тем самым связав модель с реальной вычислительной системой.

Функции, соответствующие вычисляемым атрибутам, реализуются в виде классов Java. На рис. 4 приведен пример функции, которая проверяет, что количество кондиционеров с допустимой температурой не ниже определенного порога. Результат работы функции – булево выражение.

Рассмотрим по строкам процесс задания функции:

3: Объявление константных порогов для функции.

6–9: В данных строках задается, что данная функция зависит от соседних объектов, имеющих входящие ребра.

12: Начало объявления метода для вычисления значения функции.

15–19: Вычисление числа кондиционеров с допустимой температурой не ниже определенного порога;

21: Проверка полученного значения на соответствие порогу.

ОСНОВНЫЕ ПРИНЦИПЫ РАБОТЫ СИСТЕМЫ КОНТРОЛЯ СУПЕРКОМПЬЮТЕРНОГО КОМПЛЕКСА

Построенную модель вычислительной системы необходимо «оживить» – наполнить ее реальными данными, поступающими от систем мониторинга суперкомпьютерного комплекса, т. е. сопоставить независимым атрибутам актуальные значения датчиков. Это обеспечивается разрабатываемой системой контроля суперкомпьютерного комплекса. Планируемая архитектура прототипа системы приведена на рис. 5.

«Модуль обработки входных данных» собирает значения датчиков мониторинга и передает их модулю «Логика работы с данными». Тот определяет, какие данные изменились и каким именно атрибутам в графе они должны соответствовать, после чего отдает их модулю «Графовая модель суперкомпьютера». При каждом изменении значений независимых атрибутов происходит вычисление всех связанных с ними вычисляемых атрибутов. За это отвечает модуль «Логика работы с функциями», а методы вычислений хранятся в модуле «Хранилище функций». Далее модуль «Логика работы с правилами» производит проверку выполнения правил – условий, накладываемых на состояние графа, касающихся в основном значений атрибутов. Правила хранятся в модуле «Хранилище правил».

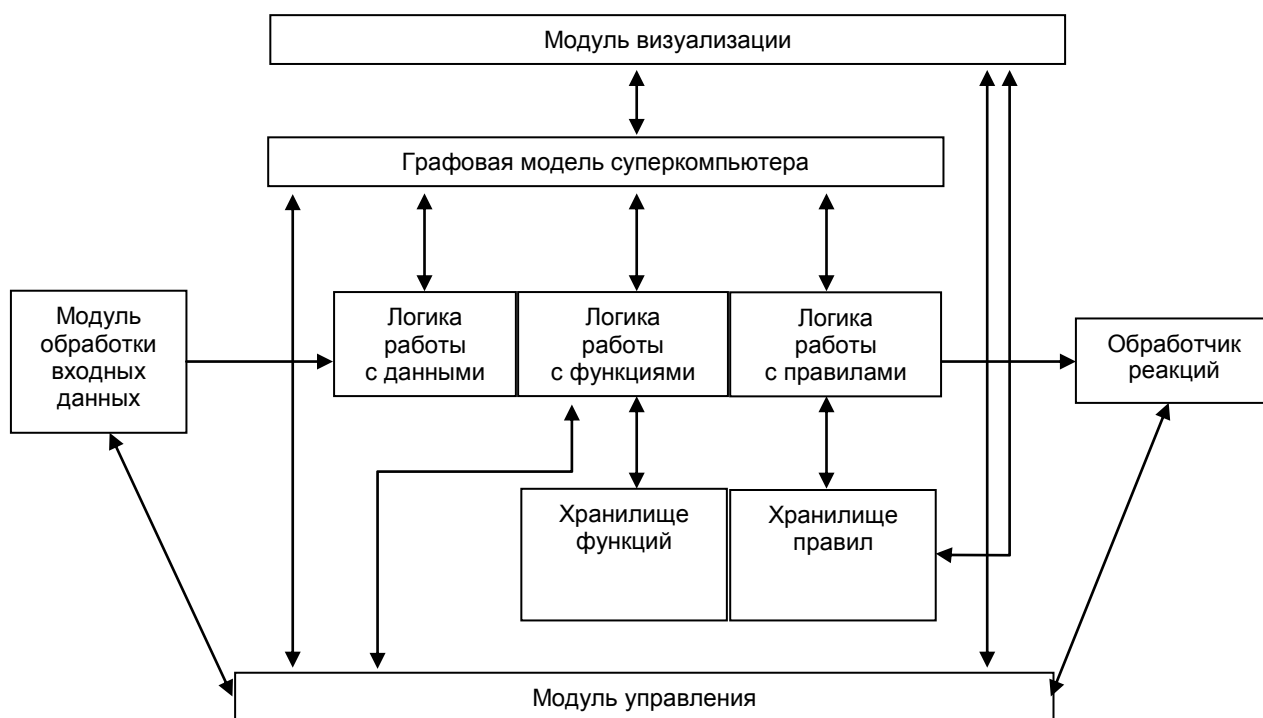


Рис. 5. Архитектура прототипа программного комплекса

В рамках правила может быть задана проверка корректного с точки зрения системы значения атрибута, в частности, его нахождение в заданных пределах. Если правило срабатывает (например, температура горячего коридора вышла за пределы допустимого порога), оно вызывает один из механизмов реакции из модуля «Обработчик реакций». Реакцией может быть оповещение ответственных лиц путем отправки SMS или e-mail, отключение питания компонентов вычислительной системы, перезагрузка вычислительных узлов.

Правила могут срабатывать не только по изменению значений атрибутов, но и по таймеру. Это необходимо для выявления ситуаций, когда неисправным может оказаться сам датчик, при этом предоставляемое им значение укладывается в заданные допуски, но со временем не изменяется.

Готовые наборы вычисляемых атрибутов и правил лягут в основу банка неисправностей и аварийных ситуаций, независимых от конкретной архитектуры вычислительного комплекса. Базовый набор таких атрибутов и правил можно использовать для адаптации системы к конкретному комплексу, при этом их можно свободно дополнять и модифицировать. Этим будет обеспечиваться требование преемственности.

Для реализации самодиагностики в системе будет предусмотрен механизм внутренних тестов, прохождение которых будет проверяться

внешними сервисами. Внутренние тесты будут основаны на тестовых датчиках, не соответствующих реальным объектам вычислительной системы, но изменяющимся по определенным законам. Набор тестов должен включать в себя:

- проверку гарантированно рабочей цепочки атрибутов (данные от тестового датчика должны меняться, чтобы происходил пересчет значений атрибутов);
- проверку цепочки атрибутов, конечный датчик которой не обновляется;
- проверку цепочки атрибутов, конечный датчик которой никогда не отдает значения;
- проверку цепочки атрибутов, конечный датчик которой периодически становится недоступным;
- проверку запуска действия по заданному значению датчика;
- проверку отработки некорректных значений датчика;
- замер времени работы рабочего цикла программы.

В качестве расширенных средств самодиагностики может быть организован независимый сбор статистики поступления информации от датчиков и анализ динамики их изменения.

Согласно проведенным тестам, графовая СУБД Neo4j и модели на ее основе обладают достаточной производительностью при размере графа, соответствующем размеру модели супер-

компьютера «Ломоносов». Для увеличения потенциала масштабируемости модель суперкомпьютерного комплекса может быть представлена как совокупность моделей, содержащих различный набор объектов и/или связей между ними. Каждая такая модель может обрабатываться независимо отдельными экземплярами системы. Более того, в случае нехватки производительности Neo4j может быть произведена миграция на другую технологическую платформу. Это требует только реализацию нужных системных API, при этом описание модели (включая атрибуты и правила) не потребует никаких изменений.

ТЕКУЩИЕ РЕЗУЛЬТАТЫ И ЗАКЛЮЧЕНИЕ

Разработка системы находится в активной стадии, поэтому некоторые описанные решения могут претерпевать существенные изменения. В частности, обсуждается вопрос об объединении понятий вычисляемых атрибутов и правил. Пока не реализованы модули визуализации и управления. В качестве тестовых средств визуализации использовался инструментарий Graphviz [12], однако с его помощью можно отрисовывать только элементарные модели. Уже сейчас ощущается нехватка средств для проверки корректности создаваемых моделей.

Нами предложен подход к описанию модели функционирования суперкомпьютерных комплексов в виде мультиграфа. Созданы программные интерфейсы для генерации моделей. Сгенерирован ряд моделей, включая модель суперкомпьютера «Чебышев». Созданы средства описания правил выявления аварийных ситуаций, с помощью которых, в частности, может быть описана логика работы системы аварийного автоматического отключения суперкомпьютера «Ломоносов».

Сформирована архитектура прототипа системы обеспечения оперативного контроля и эффективной автономной работы суперкомпьютерных комплексов. В данный момент реализуется отображение реальных данных, получаемых от системы агрегации данных мониторинга LAPTA [13–18], разработанной в рамках совместного российско-европейского проекта HOPSA [19], на модель суперкомпьютера «Чебышев». С конца 2013 г. система работает в тестовом режиме на суперкомпьютере «Чебышев». Весной 2014 г. начаты работы по запуску системы на суперкомпьютере «Ломоносов».

СПИСОК ЛИТЕРАТУРЫ

1. Система Nagios [Электронный ресурс]. URL: <http://www.nagios.org/> (дата обращения 19.12.2013). [Nagios [Online]. Available: <http://www.nagios.org/>]
2. Система Ganglia [Электронный ресурс]. URL: <http://ganglia.sourceforge.net/> (дата обращен. 19.12.2013). [Ganglia [Online]. Available: <http://ganglia.sourceforge.net/>]
3. Система Zabbix [Электронный ресурс]. URL: <http://www.zabbix.com/> (дата обращения 19.12.2013). [Zabbix [Online]. Available: <http://www.zabbix.com/>]
4. Система Zenoss [Электронный ресурс]. URL: <http://www.zenoss.org/> (дата обращения 19.12.2013). [Zenoss [Online]. Available: <http://www.zenoss.org/>]
5. Суперкомпьютерный комплекс МГУ [Электронный ресурс]. URL: <http://parallel.ru/cluster> (дата обращения 19.12.2013). [Moscow University Supercomputing Center [Online]. Available: <http://parallel.ru/cluster>]
6. Moscow University Supercomputing Center [Электронный ресурс]. URL: <http://hpc.msu.ru> (дата обращения 19.12.2013). [Moscow University Supercomputing Center [Online]. Available: <http://hpc.msu.ru>]
7. Воеводин Вл. В., Жуматий С. А., Соболев С. И., Антонов А. С., Брызгалов П. А., Никитенко Д. А., Стефанов К. С., Воеводин Вад. В. Практика суперкомпьютера "Ломоносов" // Открытые системы. 2012. № 7. [Vl. V. Voevodin, S. A. Zhumatiy, S. I. Sobolev, A. S. Antonov, P. A. Bryzgalov, D. A. Nikitenko, K. S. Stefanov, and Vad. V. Voevodin, "Practice of 'Lomonosov' Supercomputer," (in Russian), *Otkrytye Sistemy (Open Systems)*, No. 7, 2012.]
8. Sadovnichy V., Tikhonravov A., Voevodin V., Opanasenko V. "Lomonosov": Supercomputing at Moscow State University // Contemporary High Performance Computing: From Petascale toward Exascale. Boca Raton, United States: Chapman & Hall/CRC Computational Science, CRC Press, P. 283–307. [V. Sadovnichy, A. Tikhonravov, V. Voevodin, and V. Opanasenko, "'Lomonosov': Supercomputing at Moscow State University," in *Contemporary High Performance Computing: From Petascale toward Exascale*, Chapman & Hall/CRC Computational Science, CRC Press. Boca Raton, United States, pp. 283–307.]
9. Воеводин Вл. В., Жуматий С. А. Вычислительное дело и кластерные системы. М.: Изд-во МГУ, 2007. 150 с. [Vl. V. Voevodin and S. A. Zhumatiy, *Computing Work and Cluster Systems*, (in Russian). Moscow: MSU Publishing, 2007.]
10. Воеводин Вл. В., Жуматий С. А. Экспонента суперкомпьютерных центров // Открытые системы. 2008. № 5. С. 12–15. [Vl. V. Voevodin and S. A. Zhumatiy, "Exponent of supercomputing centers," (in Russian), *Otkrytye Sistemy (Open Systems)*, no. 5, pp. 12–15, 2008.]
11. Система Neo4j [Электронный ресурс]. URL: <http://www.neo4j.org/> (дата обращения 19.12.2013). [Neo4j [Online]. Available: <http://www.neo4j.org/>]
12. Система Graphviz [Электронный ресурс]. URL: <http://www.graphviz.org/> (дата обращения 19.12.2013). [Graphviz [Online]. Available: <http://www.graphviz.org/>]
13. Никитенко Д. А., Стефанов К. С. Исследование эффективности параллельных программ по данным мониторинга // Вычислительные методы и программирование. 2012. Т. 13. Раздел 2. С. 97–102 (URL: <http://nummeth.srcc.msu.ru/>). [D. A. Nikitenko and K. S. Stefanov, "Efficiency analysis of parallel programs by using of monitoring

data," (in Russian), *Numerical methods and programming: Advanced Computing*, vol. 13, pp. 97-102, 2012.]

14. **Адинец А. В., Брызгалов П. А., Воеводин Вад.В., Жуматий С. А., Никитенко Д. А., Стефанов К. С.** HOPLANG: развитие языка обработки потоков данных мониторинга // Вычислительные методы и программирование. 2012. Т. 13. Разд. 2. С. 126–131. [A. V. Adinets, P. A. Bryzgalov, Vad. V. Voevodin, S. A. Zhumatiy, D. A. Nikitenko, and K. S. Stefanov, "HOPLANG: evolution of monitoring data processing language," (in Russian), *Numerical methods and programming: Advanced Computing*, Vol. 13, pp. 126-131, 2012.]

15. **Adinets A. V., Bryzgalov P. A., Voevodin Vad. V., Zhumatiy S. A., Nikitenko D. A., Stefanov K. S.** Job Digest: an approach to dynamic analysis of job characteristics on supercomputers // *Numerical methods and programming: Advanced Computing*. 2012. Vol. 13. Section 2. P. 160–166. [A. V. Adinets, P. A. Bryzgalov, Vad. V. Voevodin, S. A. Zhumatiy, D. A. Nikitenko, and K. S. Stefanov, "Job Digest: an approach to dynamic analysis of job characteristics on supercomputers," *Numerical methods and programming: Advanced Computing*, vol. 13, Section 2, pp. 160-166, 2012.]

16. **Адинец А. В., Брызгалов П. А., Воеводин В. В., Жуматий С. А., Никитенко Д. А., Стефанов К. С.** JobDigest – подход к исследованию динамических свойств задач на суперкомпьютерных системах // Вестник УГАТУ. 2013. Т. 17, № 2 (55). С. 131–137. [A. V. Adinets, P. A. Bryzgalov, Vad. V. Voevodin, S. A. Zhumatiy, D. A. Nikitenko, and K. S. Stefanov, "Job Digest: an approach to dynamic analysis of job characteristics on supercomputers," (in Russian), *Vestnik UGATU*, vol. 17, no. 2 (55), pp. 131-137, 2013.]

17. **Брызгалов П. А., Жуматий С. А., Никитенко Д. А., Адинец А. В.** Система визуализации параметров работы больших вычислительных систем // Параллельные вычислительные технологии 2012 (ПаВТ'12): сб. тр. междунар. науч. конф. (Новосибирск, 26-30 марта 2012 г.). Челябинск: Издательский центр ЮУрГУ, 2012. С. 714. [P. A. Bryzgalov, S. A. Zhumatiy, D. A. Nikitenko, and A. V. Adinets, "Visualizing of large computing systems functioning," in *Parallel Computing Technologies 2012: Int. Conf. Proc.*, p. 714, 2012.]

18. **Адинец А. В., Брызгалов П. А., Воеводин Вад. В., Жуматий С. А., Никитенко Д. А.** Мониторинг, анализ и визуализация потока заданий на кластерной системе // Высокопроизводительные параллельные вычисления на кластерных системах: матер. XI Всерос. конф. (Н. Новгород, 1–3 нояб. 2011). Н. Новгород: Изд-во Нижегородск. гос. унта, 2011. С. 10–14. [A. V. Adinets, P. A. Bryzgalov, Vad. V. Voevodin, S. A. Zhumatiy, and D. A. Nikitenko, "Monitoring, analysis and visualizing of tasks flow on cluster system," (in Russian), in *Parallel Computing on Cluster Systems: Proc. XI Int. Conf.* (Nizhny Novgorod, Russia, Nov. 1–3, 2011), pp. 10-14, Nizhny Novgorod: Nizhny Novgorod State University Publishing, 2011.]

19. **Bernd Mohr, Vladimir Voevodin, Judit Giménez, Erik Hagersten, Andreas Knüpfer, Dmitry A. Nikitenko, Mats Nilsson, Harald Servat, Aamer Shah, Felix Wolf, Ilya Zhujov.** The HOPSA Workflow and Tools // Proc. 6th Int. Parallel Tools Workshop (Stuttgart, September 2012). Springer (to be appear). [Bernd Mohr, Vladimir Voevodin, Judit Giménez, Erik Hagersten, Andreas Knüpfer, Dmitry A. Nikitenko, Mats Nilsson, Harald Servat, Aamer Shah, Felix Wolf, and Ilya Zhujov, "The HOPSA Workflow and Tools," in *Proceedings of the 6th International Parallel Tools Workshop*, Stuttgart, September 2012, Springer (to be appear).]

ОБ АВТОРАХ

АНТОНОВ Александр Сергеевич, вед. науч. сотр. лаб. параллельных информационных технологий. Дипл. прикл. математик (МГУ, 1995). Канд. физ.-мат. наук (МГУ, 1999). Иссл. в обл. технол. параллельного программирования, мат. моделей программ, оптимизации приложений, методов стат. и дин. анализа программ, интернет-технологий, архитектур компьютеров.

ВОЕВОДИН Вадим Владимирович, науч. сотр. лаб. параллельных информационных технологий. Дипл. математик, сист. программист (МГУ, 2008). Канд. физ.-мат. наук (МГУ, 2011). Иссл. в обл. эффективности программ, локальности данных.

ВОЕВОДИН Владимир Валентинович, зам. директора. Дипл. математик (МГУ, 1984). Д-р физ.-мат. наук (МГУ, 1997). Чл.-кор. РАН (2003). Иссл. в обл. суперкомп. технологий, парал. и распредел. вычислений, преобр. и оптимиз. программ.

ЖУМАТИЙ Сергей Анатольевич, вед. науч. сотр. лаб. параллельных информационных технологий. Дипл. прикл. математик (МГУ, 1996). Канд. физ.-мат. наук (МГУ, 2005). Иссл. в обл. парал. вычислений и суперкомпьютеров.

НИКИТЕНКО Дмитрий Александрович, науч. сотр. лаб. параллельных информационных технологий. Дипл. математик, сист. программист (МГУ, 2002). Готовит дисс. о методах анализа потоков задач и приложений в суперкомпьютерных системах.

СОБОЛЕВ Сергей Игоревич, ст. науч. сотр. лаб. параллельных информационных технологий. Дипл. математик, сист. программист (МГУ, 2001). Канд. физ.-мат. наук (МГУ, 2008). Иссл. в обл. парал. и распредел. вычислений.

СТЕФАНОВ Константин Сергеевич, ст. науч. сотр. лаб. параллельных информационных технологий. Дипл. математик, сист. программист (МГУ, 2001). Канд. физ.-мат. наук (МГУ, 2007). Иссл. в обл. парал. вычислений.

ШВЕЦ Павел Артемович, программист лаб. параллельных информационных технологий. Дипл. математик, сист. программист (МГУ, 2012). Иссл. в обл. сист. программирования, парал. программирования, программирования GPU.

METADATA

Title: Securing of reliable and efficient autonomous functioning of supercomputers: basic principles and system prototype.

Authors: A. S. Antonov¹, Vad. V. Voevodin², Vl. V. Voevodin³, S. A. Zhumatiy⁴, D. A. Nikitenko⁵, S. I. Sobolev⁶, K. S. Stefanov⁷, P. A. Shvets⁸

Affiliation: Research Computing Center of Moscow State University (RCC MSU), Russia.

Email: ¹asa@parallel.ru, ²vadim@parallel.ru, ³voevodin@parallel.ru, ⁴serg@parallel.ru, ⁵dan@parallel.ru, ⁶sergeys@parallel.ru, ⁷cstef@parallel.ru, ⁸shvets.pavel.sccc@gmail.com

Language: Russian.

Source: Vestnik UGATU (scientific journal of Ufa State Aviation Technical University), vol. 18, no. 2 (63), pp. 227-236, 2014. ISSN 2225-2789 (Online), ISSN 1992-6502 (Print).

Abstract: State-of-the-art supercomputer is extremely complex, expensive and energy-saturated system. Every component of supercomputer is unreliable and can fail any time. In RCC MSU we are working on the system aimed to eliminate bad after-effects of hardware and software failures and to secure a reliable and efficient autonomous functioning of supercomputers. The system is based on the supercomputer model represented as multi-graph.

Key words: supercomputer; supercomputer reliability; supercomputer fault-tolerance; supercomputer monitoring; supercomputer damages; supercomputer failures; supercomputer autonomous functioning; supercomputer model of functioning.

About authors:

ANTONOV, Alexander Sergeevich, leading research fellow in laboratory of Parallel Computing Technologies, Research Computing Center, Moscow State University. Cand. of Tech. Sci.

VOEVODIN, Vadim Vladimirovich, research fellow in laboratory of Parallel Computing Technologies, Research Computing Center, Moscow State University. Cand. of Tech. Sci.

VOEVODIN, Vladimir Valentinovich, deputy director of Research Computing Center, Moscow State University. Dr. of Tech. Sci. Corresponding Member of Russian Academy of Sciences.

ZHUMATIY, Sergey Anatol'evich, leading research fellow in laboratory of Parallel Computing Technologies, Research Computing Center, Moscow State University. Cand. of Tech. Sci.

NIKITENKO, Dmitriy Alexandrovich, research fellow in laboratory of Parallel Computing Technologies, Research Computing Center, Moscow State University.

SOBOLEV, Sergey Igorevich, research fellow in laboratory of Parallel Computing Technologies, Research Computing Center, Moscow State University. Cand. of Tech. Sci.

STEFANOV, Konstantin Sergeevich, research fellow in laboratory of Parallel Computing Technologies, Research Computing Center, Moscow State University. Cand. of Tech. Sci.

SHVETS, Pavel Artemovich, programmer in laboratory of Parallel Computing Technologies, Research Computing Center, Moscow State University.