АВТОМАТИЧЕСКОЕ ОПРЕДЕЛЕНИЕ И ОПИСАНИЕ СЕТЕВОЙ ИНФРАСТРУКТУРЫ СУПЕРКОМПЬЮТЕРА

В.В. Воеводин, К.С. Стефанов

НИВЦ МГУ, Москва

Введение

С каждым годом наблюдается рост производительности суперкомпьютерных систем [1]. Это, в свою очередь, приводит к их усложнению – увеличивается число вычислительных узлов, усложняются иерархия подсистемы памяти и коммуникационная сеть и т. д. Все это приводит к снижению надежности и эффективности функционирования системы. Как следствие, все более актуальной становится задача обеспечения оперативного контроля и эффективной автономной работы суперкомпьютерных комплексов [2].

Для решения данной задачи в НИВЦ МГУ ведется разработка системы «Октотрон» [3,4], основная цель которой заключается в обеспечении максимальной сохранности оборудования и максимально полного его использования.

Система «Октотрон» работает на основе модели вычислительной системы, которая должна отражать основные компоненты суперкомпьютера и их взаимосвязь. Для описания моделей была разработана библиотека на языке Python, которая предоставляет простой и удобный в использовании интерфейс [5].

Однако необходимое для модели описание компонентов даже не самой большой вычислительной системы требует значительного времени и глубокого понимания ее архитектуры. Например, описание суперкомпьютера «Чебышев» [6] состоит из десятков тысяч объектов и еще большего числа связей между ними, при этом далеко не всегда связи и объекты однотипны и могут быть просто скопированы. Более того, нередка ситуация, при которой часть данных об архитектуре системы может быть по тем или иным причинам недоступна, поэтому составление подробного описания оказывается еще более сложным делом. При этом нужно стремиться с созданию как можно более полного описания, поскольку чем точнее модель, тем больше аспектов поведения суперкомпьютера удается контролировать.

Стало понятно, что процесс описания системы необходимо автоматизировать. Решено было начать с поиска программного средства для автоматического определения топологии сетей Ethernet и Infiniband в суперкомпьютере. Такая программа должна иметь доступ к целевому суперкомпьютеру, описание которого необходимо создать, и собирать доступную информацию о его строении. Мы допускаем, что подобная программа не сможет извлечь всю требуемую информацию обо всех компонентах суперкомпьютерной системы, однако основа описания должна быть получена, что позволит в значительной степени упростить процесс создания модели.

В настоящее время существует достаточно много различных систем, позволяющих определять топологию сетей вычислительных комплексов, таких как CiscoWorks Campus Manager [7], HP Network Node Manager (NNMi) [8], Netdisco [9], Zabbix [10], Nagios [11] и т. п. Однако требуемое в рамках нашего проекта программное средство должно получать по возможности наиболее полное описание архитектуры, а также быть открытым, переносимым и гибко настраиваемым. Оно должно обладать следующими необходимыми для нас свойствами:

1. определять не только коммутаторы в сети, но и вычислительные узлы;
2. обладать возможностью задания набора данных, получаемых от узлов;
3. уметь выдавать полученную информацию в требуемом для нас формате;
4. быть свободно-распространяемой.

Ни одна из рассмотренных нами утилит не удовлетворяет всем данным требованиям, поэтому было принято решение о разработке собственного программного средства для автоматического определения топологии сетей.

Далее приведено описание разработанной программы.

1. Общий алгоритм работы

Вначале будет приведено подробное описание модуля программы, предназначенного для определения топологии сети Ethernet. Модуль по определению топологии Infiniband будет описан более кратко, поскольку, во-первых, он устроен заметно проще, и, во-вторых, частично использует те же механизмы, что и первый модуль.

1.1 Определение топологии сети Ethernet

Работа модуля определения топологии сети Ethernet выполняется в два этапа. На начальном этапе производится обход всех узлов системы, с них собирается вся необходимая информация, которая затем

практически без обработки сохраняется в файлы. На следующем этапе производится анализ и преобразование всех полученных данных, которые после могут быть экспортированы в требуемом формате, в частности, в качестве описания системы для дальнейшего создания модели.

В результате такого разделения доступ к ресурсам суперкомпьютерной системы происходит только на начальном этапе, а следующий этап может быть выполнен на любой машине. В дальнейшем будем именовать эти этапы «этап сбора информации» и «этап анализа» соответственно.

Рассмотрим эти этапы подробнее.

Этап сбора информации состоит из двух шагов. На шаге 1 модуль считывает входные параметры, основным из которых является диапазон IP-адресов, который содержит все интересующие нас узлы. Блок-схема алгоритма шага 1 представлена на рис. 1.

Рис. 1. Блок-схема алгоритма работы модуля определения топологии сети Ethernet на шаге 1 этапа сбора информации.

В результате работы модуля на шаге 1 происходит обнаружение всех доступных узлов и определение их типов. Всего в модуле на данный момент выделяется три типа узлов: 1) Node – вычислительный узел; 2) Switch – коммутатор; 3) Unknown – узел неизвестного типа. Тип 3 присваивается всем узлам, для которых не удалось установить принадлежность к первым двум типам. Это может произойти вследствие недостатка собранной информации, либо узел действительно не принадлежит к первым двум типам (например, является частью системы охлаждения). Исходя из рис. 1, можно увидеть, что для корректной работы программы с головного узла (на котором производится запуск программы) вычислительные узлы должны быть доступны по протоколу SSH, а коммутаторы – по протоколу SNMP [12] и, желательно, поддерживать протокол LLDP [13] и уметь выдавать получаемую при помощи этого протокола информацию при помощи SNMP. SNMP и LLDP — стандартные сетевые протоколы, которые позволяют получать различную информацию о сетевом оборудовании и его окружении.

После определения типа узла производится попытка определения MAC-адреса, ассоциированного с данным IP-адресом, на основе данных из ARP-таблицы головного узла.

Для узлов типа Switch также по SNMP-протоколу собирается следующая информация:

1. данные, полученные этим коммутатором по протоколу LLDP;
2. список MAC-адресов сетевых интерфейсов коммутатора;
3. список IP-адресов коммутатора и IP-сетей, к которым эти адреса принадлежат;
4. список IP-адресов, которые являются следующими узлами (next hop) для всех сконфигурированных маршрутов.

Пунктирная линия на рис. 1 означает, что шаг 1 может быть при необходимости повторен требуемое число раз. Это может понадобиться в случае если топология суперкомпьютерной системы меняется в динамике. В этом случае в каждый момент времени часть узлов и связей между ними может отсутствовать, но информация о них также должна быть включена в описание системы. При следующем проходе данные об отсутствующих узлах и связях могут появиться, поэтому для получения по возможности наиболее полной картины может понадобиться несколько проходов.

Результатом работы программы на шаге 1 является файл со списком найденных узлов с указанием их IP и MAC-адресов, а также файлы с данными, полученными по протоколу SNMP.

Стоит отметить, что достаточно часто на шаге 1 не получается по IP-адресу обнаружить MAC-адрес некоторых узлов, поскольку соответствующая запись по той или иной причине отсутствовала в ARP-таблице головного узла. На устранение данного недостатка направлен шаг 2. Общий алгоритм действий на данном шаге таков:

1. собрать списки IP-сетей со всех коммутаторов, полученные на шаге 1, воедино;
2. для каждой из IP-сетей:
    1. выполнить команду ping к каждому из узлов в текущей сети;
    2. получить по SNMP ARP-таблицу со всех коммутаторов, имеющих адреса в данной сети;
    3. дополнить новыми найденными MAC-адресами записи с соответствующими IP-адресами в файле-списке найденных узлов.

Выполнение команды ping к узлам в пункте 2.1 необходимо для того, чтобы с бóльшей вероятностью записи о соответствующих узлах присутствовали в ARP-таблице коммутаторов.

На **этапе анализа** выполняется вся интеллектуальная работа программы. Данный этап также состоит из нескольких последовательно выполняемых шагов.

Вначале происходит считывание и сбор воедино всей информации, полученной на предыдущем этапе. В результате создается единая структура данных, которая содержит списки узлов каждого типа с найденной о них информацией. Коммутаторы помимо этого содержат список портов, к каждому из которых прикреплен список объектов, с которыми данный коммутатор связан по этому порту.

Затем выполняется собственно анализ и преобразование собранной информации. На этом стоит остановиться подробнее и привести описание выполняемых операций.

В основном все преобразования строятся на основе следующих положений:

1. в реальной сети один порт коммутатора может быть физически связан только с одним объектом;
2. собранные по протоколу SNMP данные о связи коммутатора с другими узлами могут указывать, что один порт коммутатора связан с множеством объектов.

Пункт 2 обуславливается следующим фактом. Указанные данные содержат информацию о том, с какими узлами этот коммутатор логически связан и через какой порт осуществляется эта связь. Под логической связью подразумевается способность «увидеть» другой узел сети через определенный порт. Частным случаем логической связи является физическая связь — прямое соединение коммутатора с узлом посредством Ethernet-кабеля. Нас интересует только физическая связь, поскольку именно эта информация нужна для определения топологии сети, поэтому на устранение остальных логических связей направлено большинство предлагаемых преобразований.

Всего производится 8 различных преобразований, которые можно условно разбить на 2 типа: конструктивные преобразования (отвечают за привнесение новой информации на основе уже имеющейся) и преобразования-«уборщики» (отвечают за удаление повторяющейся и ненужной информации). Рассмотрим их:

1. Конструктивные:
    1. *Добавление LLDP-данных*. Данные, полученные коммутатором по протоколу LLDP, содержат указание о физических связях с другими коммутаторами. Эта информация является очень ценной, поскольку определение связей между коммутаторами является, пожалуй, самой сложной задачей при определении топологи сети Ethernet. Если такая физическая связь найдена, то список объектов на соответствующем порту каждого из коммутаторов удаляется, и добавляется один объект-коммутатор, с которым была найдена связь.
    2. *Удаление лишних связей*. Если коммутатор на некотором порту логически связан с другим коммутатором, а также с несколькими узлами другого типа, то последние необходимо удалить. Это обуславливается тем фактом, что такая топология в действительности может быть устроена

единственным способом: данный коммутатор связан с другим коммутатором, который, в свою очередь, связан (на разных портах) с остальными узлами другого типа. Таким образом, физической связи между исходным коммутатором и узлами другого типа не существует, поэтому ее необходимо удалить.

3. *Поиск ближайшего коммутатора*. Если коммутатор A на некотором порту логически связан с несколькими другими коммутаторами (B_1,...,B_n), необходимо среди них найти «ближайшего соседа» — тот коммутатор, с которым установлена физическая связь. На данный момент в разработанной утилите применяется следующий алгоритм поиска. Рассматриваются все указанные коммутаторы-соседи; если у некоторого коммутатора B_i есть порт, через который он логически связан только с коммутатором A (и ни с каким другим B_j), то между коммутаторами A и B_{1i} установлена физическая связь.
4. *Обнаружение новых коммутаторов.* Если коммутатор на некотором порту логически связан с несколькими узлами другого типа, это означает, что между этими узлами и самим коммутатором есть еще один, не найденный ранее коммутатор.
5. *Добавление неизвестных узлов.* Нередки случаи, когда данные о связи коммутатора с другими узлами указывают на наличие некоторого узла с определенным MAC-адресом, который, однако, не был обнаружен ни на этапе 1, ни во время других преобразований. В этом случае создается новый объект для данного узла неизвестного типа.

2. «Уборщики»:
   1. *Удаление дополнительных MAC-адресов коммутатора*. Среди SNMP-данных может быть найден список MAC-адресов, принадлежащих одному коммутатору. На данный момент мы хотим получить общую схему топологии, и в этом случае удобнее рассматривать коммутатор как единую, неделимую сущность. Поэтому для каждого коммутатора оставляется только один MAC-адрес (остальные адреса заменяются на данный).
   2. *Удаление дополнительных IP-адресов коммутатора.* Выполняется та же самая процедура, на этот раз со списком IP-адресов коммутатора.
   3. *Удаление избыточной информации.* На данном шаге удаляются все повторяющиеся или пустые узлы и связи между ними, которые могут образоваться в результате неполноты полученных данных на этапе сбора информации или после выполнения других преобразований.

Стоит отметить, что для получения полного и корректного описания топологии важен порядок выполнения данных преобразований.

После проведения анализа и преобразования полученных данных выполняется создание графа, представляющего топологию суперкомпьютера. Полученный граф удобен для выполнения обхода всех узлов по найденным связям между ними, что используется для экспорта данных.

Полученные результаты, как и говорилось в начале статьи, экспортируются в виде скрипта на языке Python, описывающего исследуемую вычислительную систему. Далее предполагается ручная модификация полученного скрипта для уточнения полученного описания.

Также возможен экспорт описания топологии в стандартном формате dot, что позволяет получать отображение топологии сети Ethernet различными средствами визуализации, такими как, например, Graphviz [14]. Отметим, что визуальное представление сети зачастую исключительно полезно для проверки полученного описания, поэтому данному аспекту работы программы уделяется отдельное внимание.

Основная часть этапа сбора информации выполняется с помощью bash-скриптов, за исключением небольшой обвязки на языке Python на шаге 2. Этап анализа полностью реализован на языке Python.

1.2 Определение топологии сети Infiniband

Теперь перейдем к рассмотрению модуля по определению топологии сети Infiniband. Данный модуль также состоит из двух этапов, на первом из которых производится сбор необходимой информации, а на втором – анализ и преобразование полученных результатов к требуемому формату.

Этап сбора информации также состоит из двух шагов, однако они устроены заметно проще. На шаге 1 данного этапа выполняется стандартная команда ibnetdiscover, которая выдает полную информацию об Infiniband-связях в суперкомпьютере. Данная информация не позволяет определить текущее состояние связей (возможна ли в данный момент передача данных по этим связям), однако говорит о наличии связи в принципе. Таким образом, мы получаем карту всех потенциальных связей в сети Infiniband. Поскольку цель создания программы – помощь в получении описания суперкомпьютера, в которой должна присутствовать вся информация о сети Infiniband, именно такая информация нас и интересует.

Данная команда позволяет идентифицировать узлы по их GUID идентификатору. Однако во многих случаях вычислительные узлы (а иногда и коммутаторы) обладают IP-адресами. Подобная информация также представляет интерес, поэтому второй шаг заключается в нахождении ассоциации IP <-> GUID.

Выполняется это следующим образом. Вначале, как и в случае с сетью Ethernet, необходимо задать диапазон IP-адресов, где нужно выполнить поиск. Затем bash-скрипт проходит по заданному диапазону и на те

узлы, которые отвечают на команду ping, пытается зайти по ssh и выполнить команду ibstat. Если это удается, IP-адрес данного узла и ассоциированный с ним GUID записываются в выходной файл.

Этап анализа также устроен достаточно просто. Вначале считывается вся необходимая информация, полученная после выполнения команды ibnetdiscover, которая затем дополняется найденными ассоциациями IP <-> GUID.

Затем выполняется единственное на данном этапе преобразование. Некоторые корневые коммутаторы отображаются в виде нескольких составных частей, каждая из которых обладает собственным идентификатором. Поэтому необходимо вычислить, из каких частей состоит каждый коммутатор, и объединить их в единую сущность. Для этого осуществляется поиск совпадений по другому, дополнительному GUID идентификатору, информация о котором также содержится в выдаче ibnetdiscover.

Затем выполняется создание графа, полностью аналогичное соответствующему этапу в модуле по определению топологии сети Ethernet. В результате также возможен экспорт полученных данных либо в виде файла с описанием исследуемой системы, либо в виде dot-файла для дальнейшей визуализации.

2. Примеры использования

На данный момент была выполнена апробация программы в Суперкомпьютерном комплексе МГУ, а именно, на суперкомпьютерах «Чебышев» и «Ломоносов» [15] (число вычислительных узлов – около 600 и 6000, пиковая производительность — 60 Тфлопс и 1,7 Пфлопс, соответственно). В таблице 1 приведено время работы программы на различных этапах.

Таблица 1. Сравнение времени выполнения программы на суперкомпьютерах «Чебышев» и «Ломоносов»

	Этап	СК «Чебышев»	СК «Ломоносов»
Время определения топологии сети Ethernet, сек	Сбор информации	121	1846
	Анализ	< 1	350
Время определения топологии сети Infiniband, сек	Сбор информации	141	1061
	Анализ	< 1	11

Отметим, что длительное время работы на этапах сбора информации в немалой степени обусловлено необходимостью искусственного снижения потока запросов, чтобы не повлиять на работу самой системы и, в частности, не вызвать переполнение служебных таблиц при обходе большого диапазона IP-адресов.

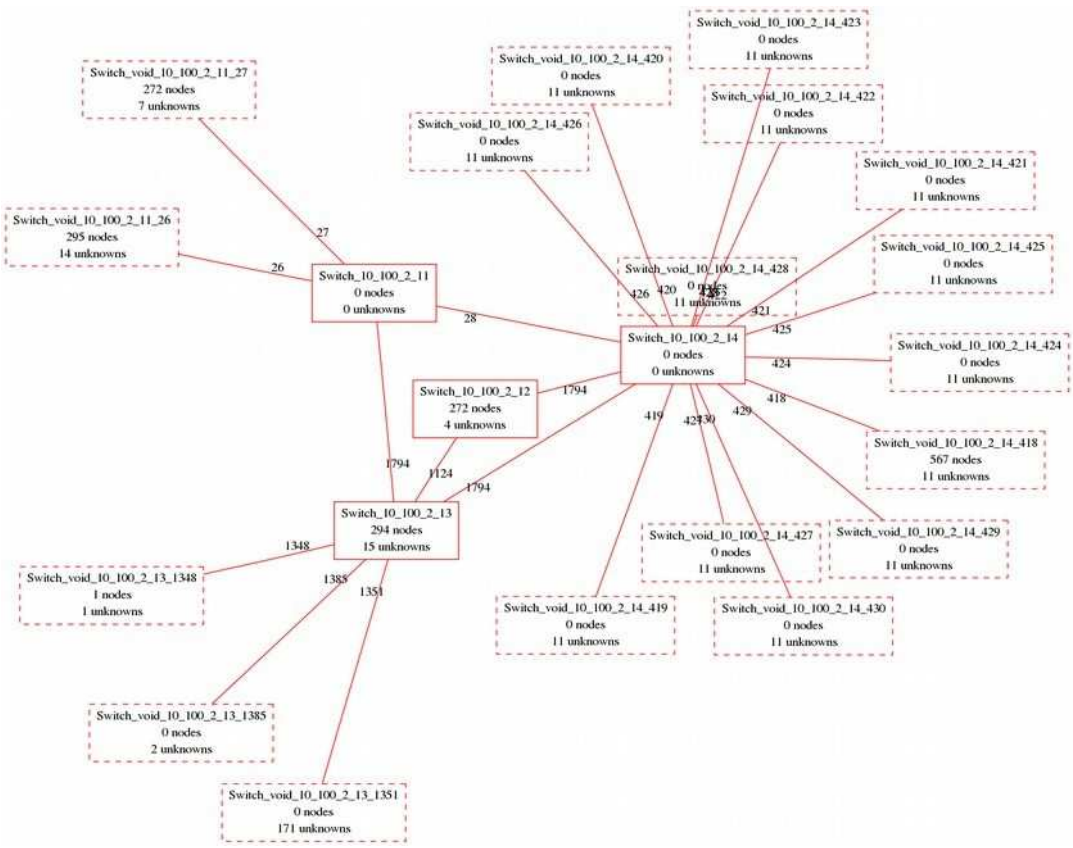

Рис. 2. Топология сети Ethernet суперкомпьютера «Чебышев»

На рис. 2 и 3 отображена соответственно топология сети Ethernet и Infiniband для суперкомпьютера «Чебышев». Изображение построено с помощью пакета Graphviz. На данных рисунках представлены только коммутаторы и связи между ними. Узлы были намеренно опущены, поскольку в противном случае рисунки становятся чрезмерно перегруженными информацией и практически нечитаемыми. В каждой вершине, соответствующей определенному коммутатору, указано число различных узлов других типов, напрямую связанных с данным коммутатором.

Пунктирными линиями выделены коммутаторы, которые не были найдены напрямую, но существование которых было обнаружено аналитически. На линиях связи указаны номера портов, на которых установлено соединение.

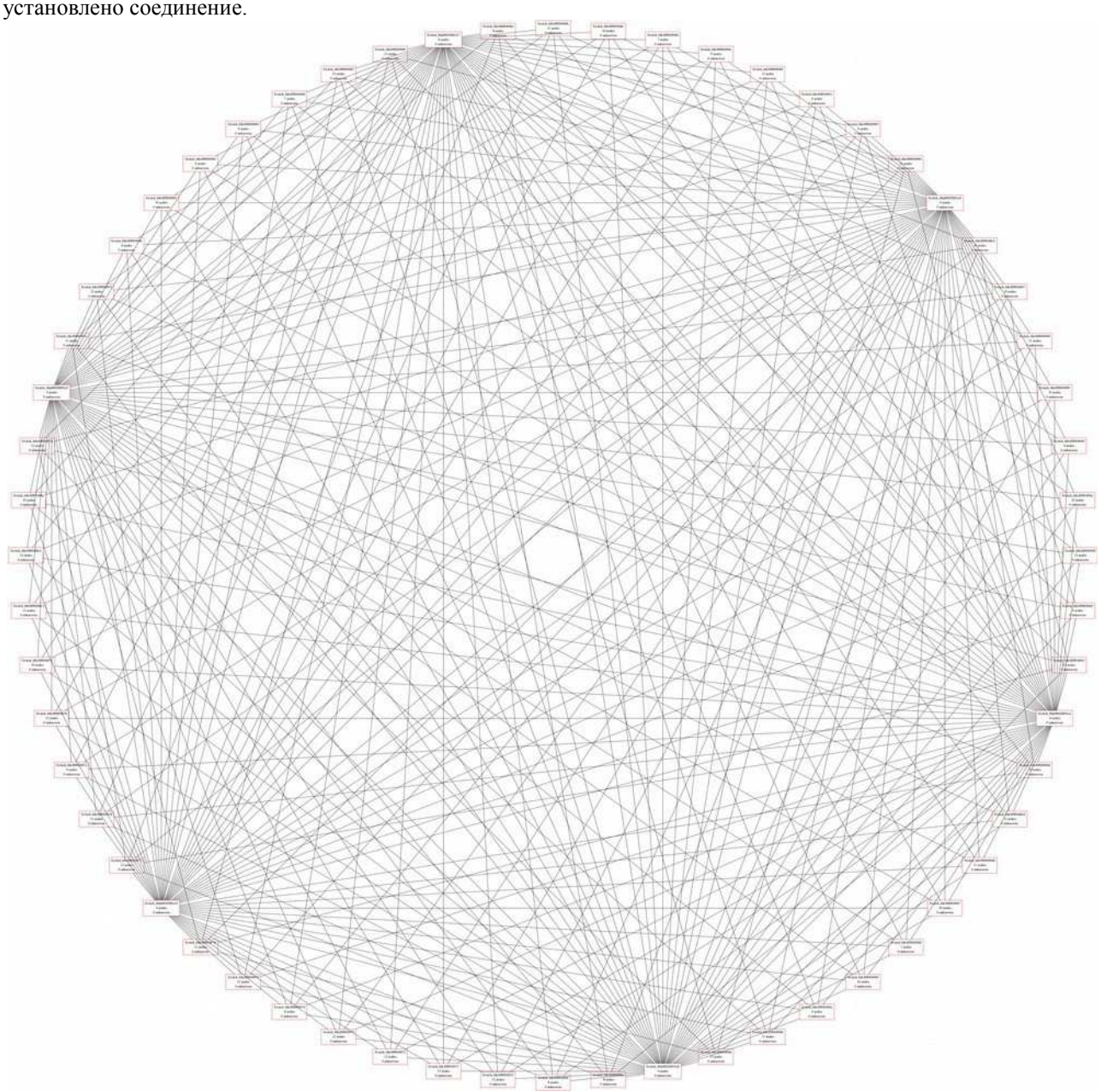

Рис. 3. Топология сети Infiniband суперкомпьютера «Чебышев»

Рис. 2 и 3 были использованы для проверки полученного описания суперкомпьютера «Чебышев», которая показала полное соответствие реальной топологии сетей.

Стоит отметить, что если в случае с суперкомпьютером «Ломоносов» аналогичное изображение топологии сети Ethernet еще может быть полезным, то изображение для сети Infiniband становится нечитаемым. В данный момент одной из основных задач на будущее является поиск подходящих средств отображения, позволяющих осуществлять удобный просмотр отдельных фрагментов топологии, а также корректную агрегацию однотипных узлов при отображении.

В результате апробации разработанной программы на системе «Чебышев» были найдены все основные коммутаторы и связи между ними как для сети Ethernet, так и для сети Infiniband. В случае системы «Ломоносов» также были найдены все коммутаторы сети Ethernet и практически все связи между ними.

Несколько связей не были обнаружены вследствие особенностей настройки некоторых коммутаторов, а также отсутствия LLDP-информации. Однако изначально предполагалось, что утилита послужит для создания основы описания, и что небольшая часть данных может в этой основе отсутствовать.

Анализ корректности и полноты полученной топологии сети Infiniband системы «Ломоносов» находится в процессе выполнения.

3. Заключение

В рамках данной работы была разработана первая версия программы по автоматическому определению топологии сетей Ethernet и Infiniband. Основной целью данной программы является существенное упрощение процесса описания суперкомпьютера. Данное описание применяется в системе обеспечения оперативного контроля и эффективной автономной работы суперкомпьютерных комплексов «Октотрон», также разрабатываемой в НИВЦ МГУ. Программа была успешно апробирована в Суперкомпьютерном комплексе МГУ: получено подробное и корректное описание сетевой инфраструктуры суперкомпьютеров «Ломоносов» и «Чебышев».

В дальнейшем планируется разработать более удобные способы визуализации топологии сетей, а также улучшить качество определения топологии за счет добавления новых методов сбора и анализа информации о вычислительной системе.

Работа выполняется при финансовой поддержке РФФИ, грант №12-07-33047.

ЛИТЕРАТУРА:

1. E. Strohmaier. Highlights of the 43rd Top500 List. Presentation at ISC'14 conference. Leipzig, Germany. URL: http://top500.org/blog/slides-for-the-43rd-top500-list-now-available/
2. А.С. Антонов, Вад В. Воеводин, С.А. Жуматий, Д.А. Никитенко, К.С. Стефанов, and П.А. Швец. Автоматизация поиска ошибок и неэффективностей в параллельных программах. Вычислительные методы и программирование: Новые вычислительные технологии (Электронный научный журнал), 14(2):11–17, 2013.
3. Антонов, А., Воеводин, В. В., Воеводин, В. В., Жуматий, С., Никитенко, Д., Соболев, С., Стефанов, К., Швец, П. Разработка принципов построения и реализация прототипа системы обеспечения оперативного контроля и эффективной автономной работы суперкомпьютерных комплексов. Вестник УГАТУ 18, 2 (2014), 227–236.
4. Исходный код текущей версии проекта «Октотрон». URL: https://github.com/srcc-msu/octotron_core
5. Рабочее окружение для создания модели на языке Python в рамках проекта «Октотрон». URL: https://github.com/srcc-msu/octotron
6. Антонов А.С. СКИФ МГУ – основа Суперкомпьютерного комплекса Московского университета// Вторая Международная научная конференция "Суперкомпьютерные системы и их применение" (SSA'2008): доклады конференции (27-29 октября 2008 года, Минск).- Минск: ОИПИ НАН Беларуси, 2008. С. 7-10.
7. Обзор продукта CiscoWorks Campus Manager 5.0. URL: http://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/ciscoworks-campus-manager-5-0/product_data_sheet0900aecd8063af4d.html
8. Обзор продукта HP Network Node Manager. URL: http://www8.hp.com/us/en/software-solutions/network-node-manager-i-network-management-software/
9. Документация по программному средству Netdisco. URL: http://www.netdisco.org/readme.html
10. Документация по программному средству Zabbix. URL: https://www.zabbix.com/documentation/ru/start
11. Документация по программному средству Nagios. URL: http://www.nagios.org/documentation
12. J. Case, M. Fedor, M. Schoffstall and J. Davin. "The Simple Network Management Protocol", STD 15, RFC 1157. May 1990. URL: http://tools.ietf.org/html/rfc1157
13. Описание протокола LLDP (формально утвержден как IEEE 802.1AB-2009). URL: http://standards.ieee.org/findstds/standard/802.1AB-2009.html
14. Документация по программному средству Graphviz. URL: http://www.graphviz.org/Documentation.php
15. Владимир Воеводин, Сергей Жуматий, Сергей Соболев, Александр Антонов, Петр Брызгалов, Дмитрий Никитенко, Константин Стефанов, Вадим Воеводин. Практика суперкомпьютера "Ломоносов"// Открытые системы, N 7, 2012. С. 36-39.