УДК 519.688

АВТОМАТИЧЕСКОЕ ОПРЕДЕЛЕНИЕ И ОПИСАНИЕ СЕТЕВОЙ ИНФРАСТРУКТУРЫ СУПЕРКОМПЬЮТЕРОВ

Вад. В. Воеводин[1], **К. С. Стефанов**[2]

С каждым годом наблюдается рост производительности суперкомпьютерных систем. Это достигается, в частности, за счет увеличения числа вычислительных узлов, усложнения иерархии подсистемы памяти и коммуникационной сети и т.д., что является одной из основных причин снижения надежности и эффективности функционирования системы. Как следствие, все более актуальной становится задача обеспечения оперативного контроля и эффективной автономной работы суперкомпьютерных комплексов. Для решения данной задачи в НИВЦ МГУ ведется разработка системы "Октотрон", 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. В данной модели должно присутствовать, в частности, описание коммуникационных сетей суперкомпьютера. Зачастую подобное описание устроено очень непросто, поэтому возникла необходимость в автоматизации этого процесса. В настоящей статье приведено описание разрабатываемого инструментария для определения топологии сетей Ethernet и Infiniband в суперкомпьютерных системах. Для построения топологии Ethernet-сети выполняется сбор SNMP-данных (Simple Network Management Protocol data) со всех доступных узлов, которые затем преобразуются на основе предлагаемого набора правил для получения более точного результата. Информация об Infiniband-сети получается на основе данных от менеджера подсети. Обсуждаются результаты работы инструментария на примере сетей суперкомпьютеров "Ломоносов" и "Чебышев", установленных в МГУ им. М. В. Ломоносова.

Ключевые слова: суперкомпьютеры, параллельные вычисления, топология суперкомпьютерных систем, коммуникационные сети, определение топологии сетей, Ethernet, Infiniband, протокол SNMP.

1. Введение. В программном средстве "Октотрон" [1, 2] требуется задать модель целевого суперкомпьютера, причем чем более точной и полной будет эта модель, тем лучше "Октотрон" сможет осуществлять свои функции. Такая модель должна включать в себя данные обо всех вычислительных узлах, сетевом и инженерном оборудовании, а также различных связях между ними.

Для описания моделей была разработана библиотека на языке Python, которая предоставляет простой и удобный в использовании интерфейс [3]. Однако необходимое для модели описание компонентов даже не самой большой вычислительной системы требует значительного времени и глубокого понимания ее архитектуры. Например, описание суперкомпьютера "Чебышев" [4] 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.

[1] Научно-исследовательский вычислительный центр Московского государственного университета им. М. В. Ломоносова, Ленинские горы, 119992, Москва; науч. сотр., e-mail: vadim@parallel.ru

[2] Научно-исследовательский вычислительный центр Московского государственного университета им. М. В. Ломоносова, Ленинские горы, 119992, Москва; ст. науч. сотр., e-mail: cstef@parallel.ru

Далее рассмотрим, какие исследования были проведены в рассматриваемой области.

Существует достаточно много работ, направленных на разработку методов определения топологии Ethernet-сетей. Выделим несколько исследований, представляющих наибольший интерес в этом отношении.

Брейтбарт и др. в своей исходной работе [5] предложили подход на основе анализа данных таблицы переадресации (AFT), получаемых по SNMP от каждого узла сети. Однако при этом накладывается серьезное ограничение, что AFT-таблица должна содержать полную информацию об узлах сети. Далее, было выяснено, что в случае наличия подсетей даже такой полной информации недостаточно для однозначного восстановления топологии. Поэтому этим коллективом авторов был предложен улучшенный подход [6], который позволяет однозначно идентифицировать определенные фрагменты сети.

В работе Лоукампа и др. [7] формулируется необходимое и достаточное условие на требуемый объем данных в AFT-таблице, который позволяет снять серьезное ограничение в виде обязательного наличия полной информации, описанное в работе [5]. Кроме того, они предполагают наличие в сети "скрытых" узлов, данные о которых могут быть найдены только в AFT-таблице (Administrative Feature Table) других узлов; напрямую эти узлы недоступны. Однако в их работе не гарантируется точное определение топологии, даже в некоторых достаточно простых случаях, как отмечено в работе [8].

Бехерано предложил [9] в 2003 г. первый формально описанный подход для обнаружения "скрытых" узлов, однако слишком большая сложность предложенного метода не позволила его реализовать. Тем не менее, эта работа позволила ему определить ограниченность применения предложенных Брейтбартом и Лоукампом и описанных выше алгоритмов [10]. В 2009 г. Бехерано представил улучшенный алгоритм и предложил вариант его реализации [11].

В 2006 г. Гобджука совместно с Брейтбартом впервые составили описание формального [12] способа определения достаточности полученных от AFT-таблиц данных для однозначного определения топологии, что послужило причиной появления серии их совместных работ по этой теме. В 2007 г. ими были выпущены работы, посвященные определению топологии при наличии "скрытых" узлов и неполной информации от AFT-таблиц [8, 13]. Кроме того, в 2007 г. они опубликовали статью с описанием алгоритма и его реализации по определению топологии при наличии "скрытых" узлов. Затем в 2010 г. ими была представлена работа с оценкой сложности определения топологии больших гетерогенных Ethernet-сетей, в которой показывается, что задача определения топологии при условии неполной информации о сети является NP-трудной [14].

Отдельно стоит упомянуть несколько работ, в которых не требуются информация из AFT-таблиц и SNMP-данные в целом. В работе Блека и др., в отличие от предыдущих работ, не используются данные от AFT-таблиц [15]. Вместо этого производится настройка узлов сети на отправку пакетов только по определенным адресам. После настройки выполняется тестовая отправка пакетов с разных узлов, и по тому, какие пакеты были доставлены, а какие нет, можно определить топологию сети. Данный метод, однако, обладает существенным недостатком: требуется установка специализированного программного обеспечения на узлы сети. Хасегава и Джибики пошли еще дальше [16]: их подход не предполагает поддержки определенной функциональности вроде SNMP-протокола, при этом вся работа по определению топологии производится с одного узла. Такой подход является гораздо более универсальным, однако приводит к снижению точности определения топологии. Более того, испытания предложенной авторами реализации проводились на не более чем 100 узлах.

Все описанные работы позволяют в значительной степени автоматизировать процесс определения топологии сети Ethernet, однако в рамках поставленной нами задачи существует ряд особенностей, которые эти работы не учитывают:

— необходимо учитывать возможное наличие VLAN-сетей (Virtual Local Area Network);

— в исследуемых Ethernet-сетях отсутствуют основные деревья (spanning tree);

— необходимо учитывать возможное изменение топологии в процессе ее построения;

— должна присутствовать возможность задания набора данных, получаемых от узлов.

Более того, не все работы привели к появлению реально существующего программного кода. Отметим, что существуют готовые коммерческие или свободно распространяемые программные системы, позволяющие определять топологию сетей вычислительных комплексов, такие как CiscoWorks Campus Manager [17], HP Network Node Manager (NNMi) [18], Netdisco [19], Zabbix [20], Nagios [21]. Однако они тоже не позволяют учесть все описанные требования.

Теперь перейдем к вопросу определения топологии сети Infiniband. Здесь ситуация кардинально отличается: в Infiniband-сети присутствует менеджер подсети (subnet manager), который отвечает за маршрутизацию пакетов. Данный сервис содержит всю необходимую информацию о топологии сети, которую

можно получить от него, например с помощью команды ibnetdiscover. В связи с этим обзор существующих решений в этой области не проводился. Таким образом, было принято решение о разработке собственного программного средства по определению топологии сетей, обладающего всей требуемой функциональностью. Далее приведено подробное описание разработанного средства.

2. 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.

2.1. 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 затем 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 (Internet Protocol address), который содержит все интересующие нас узлы. Блок-схема алгоритма шага 1 представлена на рисунке.

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: Node — вычислительный узел; Switch — коммутатор; Unknown — узел неизвестного типа. Третий тип 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). Из рисунка следует, что для корректной работы программы с головного узла (на котором производится запуск программы) вычислительные узлы должны быть доступны по протоколу SSH (Secure SHell), а коммутаторы — по протоколу SNMP [22], и желательно поддерживать протокол LLDP (Link Layer Discovery Protocol) [23] и уметь выдавать получаемую при помощи этого протокола информацию на основе протокола SNMP. Заметим, что 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 (Media Access Control address), ассоциированного с данным IP-адресом, на основе данных из ARP-таблицы (Address Resolution Protocol) головного узла.

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

— данные, полученные этим коммутатором по протоколу LLDP;

— список MAC-адресов сетевых интерфейсов коммутатора;

— список IP-адресов коммутатора и IP-сетей, к которым эти адреса принадлежат;

— список IP-адресов, которые являются следующими узлами (next hop) для всех сконфигурированных маршрутов.

Пунктирная линия на рисунке означает, что шаг 1 может быть при необходимости повторен требуемое число раз. Это может понадобиться, 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.

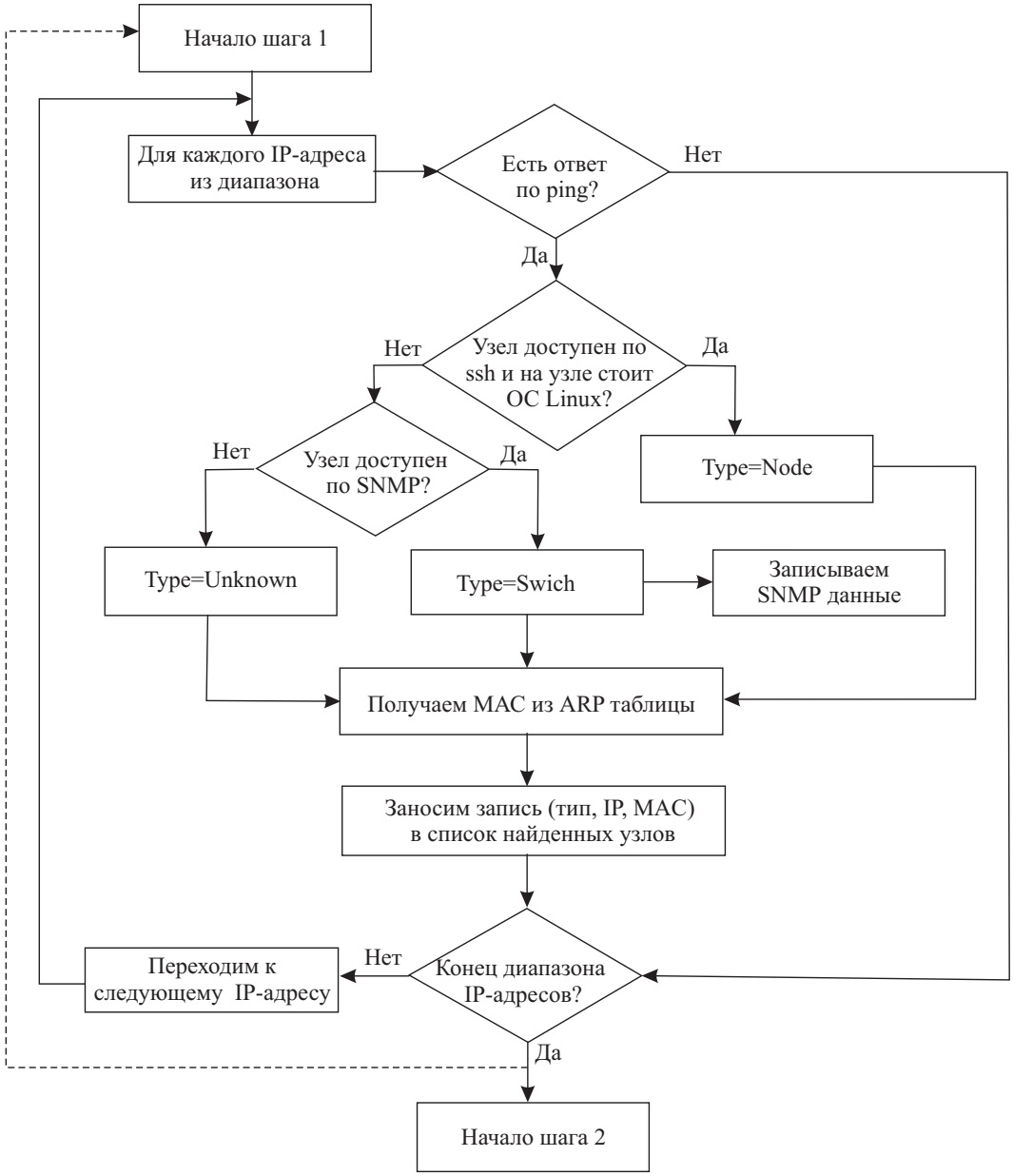
Следует отметить, что достаточно часто на шаге 1 не получается по IP-адресу обнаружить MAC-адрес некоторых узлов, поскольку соответствующая запись по той или иной причине отсутствовала в ARP-таблице головного узла. На устранение данного недостатка направлен шаг 2.

Общий алгоритм действий на втором шаге состоит следующих подшагов:

1) собрать списки IP-сетей со всех коммутаторов, полученные на шаге 1, воедино;

2) для каждой из IP-сетей:

а) выполнить команду ping к каждому из узлов в текущей сети;



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

b) получить по SNMP ARP-таблицу со всех коммутаторов, имеющих адреса в данной сети;

c) дополнить новыми найденными MAC-адресами записи с соответствующими IP-адресами в файле-списке найденных узлов.

Выполнение команды ping к узлам на подшаге 2a необходимо для того, чтобы записи о соответствующих узлах с большей вероятностью присутствовали в ARP-таблице коммутаторов.

На *этапе анализа* выполняется вся интеллектуальная работа программы. Этот 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 следует остановиться подробнее и привести описание выполняемых операций.

В основном все преобразования строятся на основе следующих положений:

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

Подшаг 2 обусловлен 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 выполняется 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 подробнее.

Конструктивные преобразования состоят в следующем.

a) *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 топологии 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.

b) *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 обусловлено 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.

c) *Поиск ближайшего коммутатора.* Если коммутатор A на некотором порту логически связан с несколькими другими коммутаторами (B_1,. . .,B_n), то 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.

d) *Обнаружение новых коммутаторов.* Если коммутатор на некотором порту логически связан с несколькими узлами другого типа, то это означает, что между этими узлами и самим коммутатором есть еще один, не найденный ранее коммутатор.

e) *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.

Преобразования-"уборщики" состоят в следующем.

a) *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).

b) *Удаление дополнительных IP-адресов коммутатора.* Выполняется та же самая процедура, на этот раз со списком IP-адресов коммутатора.

c) *Удаление избыточной информации.* На этом 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.

Следует отметить, что для получения полного и корректного описания топологии важен порядок выполнения рассмотренных 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.

Возможен экспорт описания топологии в стандартном формате dot, что позволяет получать отображение топологии сети Ethernet различными средствами визуализации, например такими, как Graphviz [24]. 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.

2.2. Определение топологии сети Infiniband. Перейдем к рассмотрению модуля по определению топологии сети Infiniband. Данный модуль тоже реализует два этапа, на первом из которых производится сбор необходимой информации, а на втором — анализ и преобразование полученных результатов к требуемому формату.

Этап сбора информации тоже 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. Эта информация не позволяет определить текущее состояние связей (возможна ли в текущий 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 котором должна присутствовать вся информация о сети Infiniband, именно такая информация нас и интересует.

Данная команда позволяет идентифицировать узлы по их GUID-идентификатору (Globally Unique IDentifier). Однако во многих случаях вычислительные узлы (а иногда и коммутаторы) обладают IP-адресами. Подобная информация так же представляет интерес, поэтому второй шаг заключается в нахождении ассоциации IP $<->$ GUID. Вначале, как и в случае с сетью Ethernet, необходимо задать диапазон IP-адресов, для которых 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, то IP-адрес данного узла и ассоциированный с ним GUID записываются в выходной файл.

Этап анализа тоже 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 возможен экспорт полученных данных либо в виде файла с описанием исследуемой системы, либо в виде dot-файла для дальнейшей визуализации.

3. Примеры использования. Была выполнена апробация программы в Суперкомпьютерном комплексе МГУ им. М. В. Ломоносова, а именно на суперкомпьютерах “Чебышев” и “Ломоносов” [25] (число вычислительных узлов — около 600 и 6000, пиковая производительность — 60 Тфлопс и 1.7 Пфлопс соответственно). В таблице приведено время работы программы на различных этапах.

Сравнение времени выполнения программы на суперкомпьютерах “Чебышев” и “Ломоносов”

Время определения топологии	Этап	СК “Чебышев”	СК “Ломоносов”
Сеть Ethernet, сек.	Сбор информации	121	1846
	Анализ	< 1	350
Сеть 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.

После определения топологии сетей они были визуализированы с помощью пакета Graphviz. Полученные изображения были использованы для визуальной проверки результатов, которая, в частности, показала, что описание суперкомпьютера "Чебышев" полностью соответствует реальной топологии сетей. Однако следует 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" тоже 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, поэтому небольшая часть данных в этом описании может отсутствовать.

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

4. Заключение. В рамках настоящей работы была разработана первая версия программы по автоматическому определению топологии сетей Ethernet и Infiniband. Основной целью этой 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 на Суперкомпьютерном комплексе МГУ им. М. В. Ломоносова: 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 выполнена при финансовой поддержке РФФИ (грант № 12–07–33047). Статья рекомендована к публикации Программным комитетом Международной суперкомпьютерной конференции "Научный сервис в сети Интернет: многообразие суперкомпьютерных миров" (http://agora.guru.ru/abrau2014).

СПИСОК ЛИТЕРАТУРЫ

1. *Антонов А.С., Воеводин Вад.В., Воеводин Вл.В., Жуматий С.А., Никитенко Д.А., Соболев С.И., Стефанов К.С., Швец П.А.* Разработка принципов построения и реализация прототипа системы обеспечения оперативного контроля и эффективной автономной работы суперкомпьютерных комплексов // Вестн. Уфимского гос. авиационного техн. ун-та. 2014. **18**, № 2. 227–236.
2. Исходный код текущей версии проекта "Октотрон" (https://github.com/srcc-msu/octotron_core).
3. Рабочее окружение для создания модели на языке Python в рамках проекта "Октотрон" (https://github.com/srcc-msu/octotron).
4. *Антонов А.С.* СКИФ МГУ — основа Суперкомпьютерного комплекса Московского университета // Вторая Международная научная конференция "Суперкомпьютерные системы и их применение" (SSA'2008). Минск: ОИПИ НАН Беларуси, 2008. 7–10.
5. *Breitbart Y., Garofalakis M., Martin C., Rastogi R., Seshadri S., Silberschatz A.* Topology discovery in heterogeneous IP networks // Proc. IEEE INFOCOM 2000. Vol. 1. New York: IEEE Press, 2000. 265–274.
6. *Breitbart Y., Garofalakis M., Jai B., Martin C., Rastogi R., Silberschatz A.* Topology discovery in heterogeneous IP networks: the *NetInventory system* // IEEE/ACM Trans. on Networking. 2004. **12**, N 3. 401–414.
7. *Lowekamp B., O'Hallaron D., Gross T.* Topology discovery for large Ethernet networks // Proc. ACM SIGCOMM 2001. San Diego: ACM Press, 2001. 237–248.
8. *Gobjuka H., Breitbart Y.* Ethernet topology discovery for networks with incomplete information // Proc. IEEE ICCCN 2007. New York: IEEE Press, 2007. 631–638.
9. *Bejerano Y., Breitbart Y., Garofalakis M., Rastogi R.* Physical topology discovery for large multisubnet networks // Proc. IEEE INFOCOM 2003. Vol. 1. New York: IEEE Press, 2003. 342–352.
10. *Bejerano Y.* Taking the skeletons out of the closets: a simple and efficient topology discovery scheme for large multisubnet networks // Proc. IEEE INFOCOM 2006. New York: IEEE Press, 2006. 1–13.
11. *Bejerano Y.* Taking the skeletons out of the closets: a simple and efficient topology discovery scheme for large Ethernet LANs // IEEE/ACM Trans. on Networking. 2009. **17**, N 5. 1385–1398.

12. *Breitbart Y., Gobjuka H.* Characterization of layer-2 unique topologies // Information Processing Letters. 2008. **105**, N 2. 52–57.
13. *Gobjuka H., Breitbart Y.* Finding Ethernet-type network topology is not easy. Technical Report N TR-KSU-CS-2007-03. Kent: Kent State Univ., 2007.
14. *Gobjuka H., Breitbart Y.* Ethernet topology discovery for networks with incomplete information // IEEE/ACM Trans. on Networking. 2010. **18**, N 4. 1220–1233.
15. *Black R., Donnelly A., Fournet C.* Ethernet topology discovery without network assistance // Proc 12th IEEE Int. Conf. on Network Protocols (ICNP 2004). New York: IEEE Press, 2004. 328–339.
16. *Hasegawa Y., Jibiki M.* Ethernet topology detection from a single host without assistance of network nodes or other hosts // IEICE Trans. on Communications. 2009. **92**, N 4. 1128–1136.
17. Обзор продукта CiscoWorks Campus Manager 5.0 (http://www.cisco.com/c/en/us/products/collateral/cloud-systems- management/ciscoworks-campus-manager-5-0/product_data_sheet0900aecd8063af4d.html).
18. Обзор продукта HP Network Node Manager (http://www8.hp.com/us/en/software-solutions/network-node-manager-i-network- management-software/).
19. Документация по программному средству Netdisco (http://www.netdisco.org/readme.html).
20. Документация по программному средству Zabbix (https://www.zabbix.com/documentation/ru/start).
21. Документация по программному средству Nagios (http://www.nagios.org/documentation).
22. *Case J., Fedor M., Schoffstall M., Davin J.* The simple network management protocol. STD 15, RFC 1157 (http://tools.ietf.org/html/rfc1157).
23. Описание протокола LLDP (формально утвержден как IEEE 802.1AB-2009) (http://standards.ieee.org/findstds/standard/802.1AB-2009.html).
24. Документация по программному средству Graphviz (http://www.graphviz.org/Documentation.php).
25. *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. 36–39.

Поступила в редакцию
17.08.2014

Automatic Detection and Description of Supercomputer Network Infrastructure

Vad. V. Voevodin [1] **and K. S. Stefanov** [2]

[1] *Research Computing Center, Lomonosov Moscow State University; Leninskie Gory, Moscow, 119992, Russia; Ph.D., Scientist, e-mail: vadim@parallel.ru*

[2] *Research Computing Center, Lomonosov Moscow State University; Leninskie Gory, Moscow, 119992, Russia; Ph.D., Senior Scientist, e-mail: cstef@parallel.ru*

Received August 17, 2014

Abstract: The supercomputing system performance increases each year. In particular, this is because of increasing the number of computational nodes, making the memory subsystem and communication network more complex, etc., which causes the reduction of reliability and system's efficiency. As a result, the on-line control and efficient autonomous functioning of supercomputer complexes become more and more important. In order to solve this problem, Moscow University Research Computing Center is currently developing the Octotron system whose main objective is to provide the maximum safety and fullest usage of the hardware. The Octotron system uses a model of a computer system that should contain main supercomputer components and their interconnection. In particular, this model should contain a description of communication networks. Such a description could be significantly complex in many cases; therefore, the automation of this process is needed. In this paper we describe a programming tool being developed in order to detect Ethernet and Infiniband network topology in supercomputer systems. For detecting Ethernet network topology, this tool collects SNMP data from all available nodes and modifies them on the basis of the proposed rules to achieve more precise results. In the case of Infiniband network, it collects the necessary information from the subnet manager. The results of using this tool on the Lomonosov and Chebyshev supercomputers installed at Moscow University are discussed.

Keywords: supercomputers, parallel computing, supercomputer topology, communication networks, network topology detection, Ethernet, Infiniband, SNMP protocol.

References

1. A. S. Antonov, Vad. V. Voevodin, Vl. V. Voevodin, et al., "Securing of Reliable and Efficient Autonomous Functioning of Supercomputers: Basic Principles and System Prototype," Vestn. Ufa Aviatsion. Tekh. Univ. **18** (2), 227–236 (2014).

2. Source Codes of the Octotron System. https://github.com/srcc-msu/octotron_core. Cited August 11, 2014.

3. Creation of Models on Python in the Octotron System. https://github.com/srcc-msu/octotron. Cited August 11, 2014.

4. A. S. Antonov, "SKIF MSU is a Basis for the Supercomputing System of Moscow State University," in *Proc. 2nd Int. Conf. on Supercomputer Systems and Applications, Minsk, Belarus, October 27–29, 2008* (United Inst. Informatics Problems, Minsk, 2008), pp. 7–10.

5. Y. Breitbart, M. Garofalakis, C. Martin, et al., "Topology Discovery in Heterogeneous IP Networks," in *Proc. 19th Conf. on Computer Communications, Tel Aviv, Israel, March 26–30, 2000* (IEEE Press, New York, 2000), Vol. 1, pp. 265–274.

6. Y. Breitbart, M. Garofalakis, B. Jai, et al., "Topology Discovery in Heterogeneous IP Networks: The *NetInventory* System," IEEE/ACM Trans. Networking **12** (3), 401–414 (2004).

7. B. Lowekamp, D. O'Hallaron, and T. Gross, "Topology Discovery for Large Ethernet Networks," in *Proc. ACM SIGCOMM 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication, San Diego, USA, August 27–31, 2001* (ACM Press, San Diego, 2001), pp. 237–248.

8. H. Gobjuka and Y. Breitbart, "Ethernet Topology Discovery for Networks with Incomplete Information," in *Proc. 16th Int. Conf. on Computer Communications and Networks. Honolulu, USA, August 13–16, 2007* (IEEE Press, New York, 2007), pp. 631–638.

9. Y. Bejerano, Y. Breitbart, M. Garofalakis, and R. Rastogi, "Physical Topology Discovery for Large Multisubnet Networks," in *Proc. 22nd Joint Conf. of the IEEE Computer and Communications Societies, San Francisco, USA, April 1–3, 2003* (IEEE Press, New York, 2003), Vol. 1, pp. 342–352.

10. Y. Bejerano, "Taking the Skeletons out of the Closets: A Simple and Efficient Topology Discovery Scheme for Large Multisubnet Networks," in *Proc. 25th Conf. on Computer Communications, Barcelona, Spain, April 23–29, 2006* (IEEE Press, New York, 2006), pp. 1–13.

11. Y. Bejerano, "Taking the Skeletons out of the Closets: A Simple and Efficient Topology Discovery Scheme for Large Ethernet LANs," IEEE/ACM Trans. Networking **17** (5), 1385–1398 (2009).

12. Y. Breitbart and H. Gobjuka, "Characterization of Layer-2 Unique Topologies," Inform. Process. Lett. **105** (2), 52–57.

13. H. Gobjuka and Y. Breitbart, *Finding Ethernet-Type Network Topology is Not Easy*, Tech. Report TR-KSU-CS-2007-03 (Kent State Univ., Kent, 2007).

14. H. Gobjuka and Y. Breitbart, "Ethernet Topology Discovery for Networks with Incomplete Information," IEEE/ACM Trans. Networking **18** (4), 1220–1233 (2010).

15. R. Black, A. Donnelly, and C. Fournet, "Ethernet Topology Discovery without Network Assistance," in *Proc 12th IEEE Int. Conf. on Network Protocols (ICNP 2004), Berlin, Germany, October 5–8, 2004* (IEEE Press, New York, 2004), pp. 328–339.

16. Y. Hasegawa and M. Jibiki, "Ethernet Topology Detection from a Single Host without Assistance of Network Nodes or Other Hosts," IEICE Trans. on Commun. **92** (4), 1128–1136 (2009).

17. CiscoWorks Campus Manager 5.0. http://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/ciscoworks-campus-manager-5-0/product_data_sheet0900aecd8063af4d.html. Cited August 11, 2014.

18. HP Network Node Manager. http://www8.hp.com/us/en/software-solutions/network-node-manager-i-network-management-software. Cited August 11, 2014.

19. Netdisco User Guide. http://www.netdisco.org/readme.html. Cited August 11, 2014.

20. Zabbix User Guide. https://www.zabbix.com/documentation/ru/start. Cited August 11, 2014.

21. Nagios User Guide. http://www.nagios.org/documentation. Cited August 11, 2014.

22. J. Case, M. Fedor, M. Schoffstall, and J. Davin, "The Simple Network Management Protocol. STD 15, RFC 1157," http://tools.ietf.org/html/rfc1157. Cited August 11, 2014.

23. IEEE Standard 802.1AB-2009. http://standards.ieee.org/findstds/standard/802.1AB-2009.html. Cited August 11, 2014.

24. Graphviz User Guide. http://www.graphviz.org/Documentation.php. Cited August 11, 2014.

25. Vl. V. Voevodin, S. A. Zhumatii, S. I. Sobolev, et al., "Practice of Lomonosov Supercomputer," Otkrytye Sistemy, No. 7, 36–39 (2012).