ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ № 3(3), 2010, с. 93–106

УДК 519.687.4

А. А. Кузнецов, В. А. Роганов

Поддержка топологии вычислительного пространства в системе OpenTS

Аннотация. Эффективное комплексирование массы разрозненных компьютеров и суперкомпьютеров требует использования технологий динамического распараллеливания, автоматической динамической реконфигурации счетного поля, динамической балансировки нагрузки, учета неоднородностей и топологии сети, а также поддержки отказоустойчивости с учетом степени надежности узлов. В статье рассматривается отечественная программная технология, разработанная под указанные выше требования в рамках программы СКИФ-ГРИД.

Ключевые слова и фразы: облачные вычисления, распределенные вычисления, GRID-системы, динамическое распараллеливание программ, Т-система с открытой архитектурой, отказоустойчивость.

1. Система программирования OpenTS

OpenTS — это система параллельного программирования, разработанная в ИПС РАН в 2000-2004 годах в рамках суперкомпьютерного проекта "СКИФ"Союзного государства России и Беларуси. После успешного завершения программы "СКИФ"разработка системы OpenTS [1, 4, 5] была продолжена в рамках программы "СКИФ-ГРИД"и различных академических проектов.

Система OpenTS (Open T-System) представляет собой современную и наиболее удачную реализацию концепции Т-системы — программной среды параллельного программирования с поддержкой динамического распараллеливания Т-приложений, которая сочетает в себе функциональную и императивную парадигмы программирования. OpenTS — это среда поддержки исполнения приложений, написанных на языке Т++. Данный язык программирования является параллельным функциональным расширением Си++ и дополняет исходный язык всего семью новыми ключевыми словами. Среда поддержки исполнения Т++ приложений (Т-приложений), берет на себя

основную часть работы по организации параллельного счета (синхронизация, распределение нагрузки, транспортировка сообщений). Тем самым, система OpenTS позволяет снизить затраты на разработку параллельных программ, увеличить глубину параллелизма и более полно использовать возможности аппаратной части мультипроцессора за счет распараллеливания в динамике.

2. Система облачных вычислений SkyTS

Большие вычислительные системы вынужденно являтся распределенными. Хотя первым применением для OpenTS и были кластерные вычисления, тем не менее ни у кого не было сомнений в том, что великолепную отдачу от T-системы (по отношению к конкурирующим технологиям) можно получить и в распределенных средах. Поэтому практически сразу появилась поддержка метакластеров, затем в транспортную подсистему DMPI была добавлена поддержка работы по TCP/IP (режим "хризантема"), добавлена прозрачная (не требующая модификации программ) отказоустойчивость [2] к сбоям вычислительных узлов и сетевого оборудования. Также, реализовано свойство масштабируемости [6], которое позволяет T-приложениям, созданным с использованием системы OpenTS, работать параллельно в сверхбольших кластерных и распределенных средах с практически неограниченным числом узлов.

Проведенные в последние годы доработки системы OpenTS (улучшение свойств кроссплатформенности, разработка подсистем отказоустойчивости и масштабируемости) позволили системе перейти на новый уровень развития, который состоит в поддержке выполнения параллельных T-приложений в неоднородной распределенной вычислительной среде. В рамках программ Союзного государства "Триада"и "СКИФ-ГРИД"разработана архитектура и проведена экспериментальная реализация распределенной отказоустойчивой системы "SkyTS"[6], которая позволяет объединить разнородные ресурсы компьютеров в сети Интернет для счета T-приложений, решающих ресурсоемкие научно-прикладные задачи. SkyTS — управляющая надстройка над OpenTS для глобальных гетерогенных распределенных сред, реализованная на простом командном языке TCL. Включает в себя также пользовательский Web-интерфейс для практичного управления запуском множества T-приложений.

3. "Планетарный"режим работы T-системы

Технологическая база развивается очень быстро. Еще недавно были выпущены модели больших отечественных суперкомпьютеров производительностью в единицы терафлопс, а новые технологии СКИФ (четвертый ряд супервычислителей "СКИФ-Аврора"с жидкостным охлаждением) уже позволяют уложить несколько терафлопс в один компактный вычислитель размером с системный блок персонального компьютера.

Но сколько бы ни было вычислительных кластеров и спецвычислителей, их, с точки зрения T-системы, всегда можно рассматривать как глобальный метакомпьютер с единым, хоть и не вполне однородным информационным пространством — общей суперпамятью. Увы, такой "планетарный"суперкомпьютер будет крайне ненадежен и весьма изменчив по достаточно очевидным причинам, так что традиционные подходы к его программированию попросту неприменимы.

Одной из ключевых идей программы СКИФ-ГРИД как раз и является такое прозрачное объединение всех суперкомпьютеров для наиболее эффективного использования имеющихся вычислительных мощностей. Система SkyTS в связке с режимом OpenTS/PlaneT как раз и обеспечивает подобный режим работы для всей совокупности как суперкомпьютеров, так и отдельных недозагруженных компьютеров.

В этой статье рассказывается о наиболее "глобальном"аспекте T-системы/OpenTS, а именно "планетарном"режиме работы "PlaneT— режиме облачных вычислений с учетом топологии облаков. В этом режиме вычислительным пространством может эффективно служить практически сколь угодно большое множество ЭВМ.

4. Топология вычислительного пространства

Под термином "топология"подразумевается определение предполагаемой сетевой связности узлов с указанием списка произвольных атрибутов у подмножеств вычислительных узлов — "вычислительных облаков". В режиме счета с учетом топологии задаются подсети ("вычислительные облака"). Вычислительные облака позволяют компактно задавать большое количество связанных в единый сегмент сети узлов, и устроены подобно классам сетей TCP/IP. Можно

считать, что топология вычислительного пространства у обычного кластера тривиальна — это попросту полносвязный граф.

Присутствие топологии важно тем, что в условиях большого (несколько тысяч и более) количества узлов в распределенной системе у любого из них всегда есть доступ к более-менее правдоподобной "информационной карте местности на которой обозначены не только каналы связи, но также ресурсы и свойства вычислительных узлов. Такая информация позволяет действовать группам узлов автономно, а также оперативно находить наиболее подходящую замену отказавшим узлам (в случае необходимости переповтора T-подзадач в случае утраты связи с нужным узлом, что может устанавливаться по факту отсутствия "сердцебиения— периодически посылаемого контрольного пакета).

4.1. Пример определения топологии

Топология представляется в виде текстового файла, который обрабатывается средствами открытой мультиплатформенной технологии Flex/Bison. Формат файла поддерживает директивы include, что удобно для объединения его фрагментов воедино. Файл с топологией может быть доступен по протоколу HTTP с помощью CGI-сценария на Web-сервере, который способен генерировать файл описания топологии для некоторой страны или региона, пользуясь данными GeoIP-системы. Пример файла определения топологии:

```
OpenTS.Net 193.232.174.43 23.34.56.78 :
# PZ network
127.0.0.1 0.7:192.168.1.2
127.0.0.1 0.3 127.0.0.1 0.6 : 192.168.1.2(public)
# Some PZ home subnetwork
# OpenTS-enabled subnetwork with leader without OpenTS
127.0.0.1(not) : 192.168.1.2 0.7(public)
# skif.botik.ru frontend as 3x2x2 torus (with it's own dmpi subtransport)
193.232.174.28 : Torus {
  { {10.0.0.1 10.0.0.2}  {10.0.0.3 10.0.0.4}   }
  { {10.0.0.5 10.0.0.6}  {10.0.0.7 10.0.0.8}   }
  { {10.0.0.9 10.0.0.10} {10.0.0.11 10.0.0.12} }
}
# T-fan's computer with 4 running OpenTS workers, emulating 4-node cluser:
12.23.45.67 0.9 {    # multi-line variation for leadered subnet definition
  127.0.0.1(port=3301)
  127.0.0.1(port=3302)
  127.0.0.1(port=3303)
  127.0.0.1(port=3304)
```

```
}
# Yet another T-computist with tiny cluster
12.23.45.68  {10.0.0.9 10.0.0.10 10.0.0.11 10.0.0.12}
# Some Moscow network
127.0.0.1 : 192.168.1.2/26(public)
127.0.0.1 0.2 : 192.168.1.2/21 0.7 (public)
# Some Moscow friend's home subnetwork
127.0.0.1 0.7 { 192.168.1.2/20(public) }
# Include yet another cluster with its frontend
56.87.78.68: include T500.3d
include http://www.opents.net/region.cgi\?name=Ivanovo
```

4.2. Поддержка обменов сообщениями напрямую между узлами распределенной сети

Подсистема Суперпамяти в OpenTS представляет собой единое адресное пространство. Это позволило в короткие сроки реализовать в подсистеме DMPI такую схему обменов сообщениями, при которой обмен происходит напрямую между узлами сети, минуя центральный узел Grid-системы [3]. Если какая-либо T-функция, порожденная на узле N, напрямик захочет выполниться на узле M (между которыми никакой особой связи не установлено), то это беспрепятственно произойдет. Для этого будет установлен публичный IP-лидер узла M и через него за два DMPI-хопа активное сообщение будет доставлено по назначению. Ответ уйдет или тем же путем, или за два DMPI-хопа через соответствующего публичного IP-лидера узла N.

Все подобные соединения устанавливаются лениво, то есть по требованию. Но с соседями по квази-иерархии и с публичными IP-лидерами (где может и не работать OpenTS) обмен сообщениями по сети происходит в обычном (энергичном) режиме. Поэтому вся эта многомиллиардная потенциальность соединений не занимает места даже в виртуальном адресном пространстве T-процессов. Таким образом организована эффективная поддержка до нескольких десятков миллионов узлов.

4.3. Мультитранспорт в подсистеме DMPI

С понятием топологии тесно связано понятие мультитранспорта (возможности DMPI работать по разным протоколам в различных сегментах сети). Топология не подменяет собой идею динамического определения свойств сети и узлов, но позволяет начинать реальный счет мгновенно, не тратя время на выявление свойств и взаимное

приветствие узлов, что в большой распределенной сети может занимать значительное время. Топология не является статической, и от запуска к запуску может меняться, в том числе и на основе апостериорной информации (например, атрибуты надежности узлов могут пересчитываться на основании статистики их сбоев). Топология может считываться как из локальных файлов, так и по сети.

Присутствие топологии важно тем, что в условиях большого (сотни тысяч и более) количества узлов в распределенной системе у любого из них всегда есть доступ к более-менее правдоподобной "информационной карте местности на которой обозначены не только каналы связи, но также ресурсы и свойства вычислительных узлов. Такая информация позволяет действовать группам узлов автономно, а также оперативно находить наиболее подходящую замену отказавшим узлам (в случае необходимости переповтора T-подзадач в случае утраты связи с нужным узлом, что может устанавливаться по факту отсутствия "сердцебиения— периодически посылаемого контрольного пакета).

5. Алгоритм работы DMPI с учетом топологии

Алгоритм работы подсистемы DMPI с учетом топологии вполне прост. Подсистема DMPI на старте;

- считывает и очень быстро компилирует априорное определение топологии;
- находит в ней свое уникальное место (узел, где она запущена);
- активирует пассивное прослушивание соответствующих сетевых интерфейсов:
- с обработкой сигналов от них в случае поступления соединений,
- с использованием асинхронного режима установки пассивных соединений;
- запрашивает у вышестоящего по квази-иерархии свой порядковый номер и список имеющихся работоспособных соседей;
- посылает асинхронные запросы на соединение к непосредственным соседям по квази-иерархии (асинхронный режим установки активных соединений).

В том случае, если узлу не удалось однозначно и достоверно отнести себя ни к одному из перечисленных в топологии узлов, он обращается к одному из так называемых лидеров квази-иерархии с просьбой назначить ему место динамически.

6. синхронность и транзиты через узлы с публичными IP-адресами

В подсистеме DMPI все коммуникации между узлами распределенной сети происходят асинхронно, что позволяет получить большой прирост производительности по отношению к синхронной модели пересылки данных, на которой склонен реализовывать программы человек. Дело в том, что в случае, когда все коммуникации асинхронны, ОС и драйверы сетевых интерфейсов используют свободу в очередности ввода-вывода. Также, в этом случае не нужно осуществлять лишнее копирование данных в память ОС, как в случае синхронных коммуникаций. Весь ввод-вывод (соединения, разъединения, авторизация, обмены данными) производится исключительно при помощи асинхронных вызовов aio_read()/aio_write() и RealTime-сигналов. Какие-либо блокирующие операции отсутствуют. Для ОС Windows на основе Win32 API разработана собственная версия программной библиотеки, которая имплементирует функции асинхронного взаимодействия aio_read()/aio_write().

В OpenTS используются "барабаны в которых последовательно помещаются десятки запросов ввода-вывода одновременно, и которые "поворачиваются"по мере их реального завершения. Ограниченная емкость барабанов, в свою очередь, не позволяет перегрузить ОС асинхронными операциями ввода-вывода. Асинхронны и транзиты, которые возникают, например, при обмене по TCP/IP между узлами из двух приватных подсетей через узел с публичным IP. Транзиты уровня активных сообщений сосуществуют с маршрутизацией уровня TCP/IP: последний функционирует сам по себе, но на более низком уровне. Транзиты уровня DMPI обычно имеют не более двух хопов, и совершаются с учетом сетевой топологии.

Для улучшения производительности транспортной подсистемы разработан предобработчик сообщений, задача которого сократить время простоя (Idle time) ядра OpenTS при работе в отказоустойчивом асинхронном режиме. Предобработчик вызывается каждый раз при получении сообщения от другого узла-участника счета, и если ядро OpenTS в этот момент не занято счетом, то сообщение сразу же обрабатывается.

6.1. Использование модели квази-иерархии гиперкубов

Модель квази-иерархии гиперкубов использована в системе OpenTS при реализации подсистемы масштабируемости T++ приложений. Под квази-иерархией подразумевается используемая в настоящий момент структура, состоящая из гиперкубов с ребром длины один. Такой гиперкуб можно представить как множество полей бит определенной длины. Например, все возможные шестерки бит являются вершинами шестимерного гиперкуба, их всего 64. Гиперкубы образуют уровни квази-иерархии следующим образом: все вершины гиперкуба — это уровни (соседи), вершины, битовое представление которых отличается ровно N битами — непосредственные соседи (например, в случае N=1 у каждой вершины шестимерного гиперкуба будет шесть непосредственных соседей). У каждой вершины может быть один или несколько подуровней (вложенных гиперкубов), а также один или несколько родителей.

Гиперкубы хороши тем, что если вершин не хватает для их заполнения, то начальные заполнения также являются гиперкубами, просто меньшей размерности. То же самое верно и для любой из граней — они также являются гиперкубами. При разбиении задачи на подзадачи по квази-иерархии сверху вниз количество элементов на одном уровне становится коэффициентом размножения. Например, при использовании шестимерных гиперкубов коэффициент размножения равен 64. За три уровня задача разбивается на 260 тысяч, за пять — на миллиард частей. Вертикально-горизонтальные связи избыточны и позволяют проводить динамическую реконфигурацию при возникновении сбоев без остановки счета.

Квази-иерархия гиперкубов возникла из задачи масштабирования, где пришлось минимизировать количество связей между узлами в случае огромного их количества. В настоящее время реализация суперпамяти позволяет адресовать до 2-х миллиардов супер-ячеек (распределенных элементов данных) на 2-х миллиардах узлов.

Память под супер-ячейки выделяется динамически — по требованию. Это позволяет, с одной стороны, сохранить единое адресное пространство (то есть, любой узел может непосредственно адресовать супер-ячейки каждого), но, с другой стороны, в нормальной ситуации таких связей не требуется, а следовательно, и не возникает. Поэтому и ресурсов под эти "бесконечные потенциальные"связи заранее не выделяется. Другое дело — непосредственные узлы-соседи. С ними

связь устанавливается одновременно и немедленно. Также устанавливается связь с так называемым "публичным лидером". Это транзитный механизм, который как раз и нужен для инициации непредвиденных потенциальных связей, которые упоминались выше.

Гиперкубы взаимоувязаны с топологией следующим образом: если на каком-то уровне квази-иерархии получается слишком много узлов, они автоматически дробятся по гиперкубам.

6.2. Отказоустойчивость

Иерархичность вызовов Т-функции и используемая схема отказоустойчивости гарантирует выполнение задачи, пока есть соответствующие возможности. Такая задача разбивается на подзадачи, между которыми возникают связи. Неготовые значения позволяют асинхронно запускать несколько независимых активностей.

Отказоустойчивость реализована на базе модели перевычислений с использованием т. н. порталов. Они нужны для фиксации переходов Т-подзадач из одного вычислительного подпространства в другое. В случае потери связи с узлом или выхода последнего из строя портал содержит в точности те Т-функции, которые нужно перевычислить.

Промежуточные узлы квази-иерархии не только что-то считают, но и управляют дальнейшими уровнями. Планировщик старается пересылать промежуточные функции на промежуточные узлы, а на роль последних выдвигает наиболее надежные (по указанному значению в атрибуте "надежность"). Тем не менее, и промежуточные узлы могут выйти из строя. Если выведен из строя промежуточный узел квази-иерархии, то его подчиненные узлы пытаются найти свое место у следующего вышестоящего лидера и т д.

Но что делать с подзадачами, которые они, возможно, уже переправили далее? Для автоматического останова "бесхозных"подзадач (если утрачен породивший их узел) используются понятия вектора перерождений и вектора посещений. Вектор перерождений для каждого вычислительного узла содержит номер перерождения (перезагрузки) и статус (0 — не был доступен; 1 — первый раз стал доступен; 2 — недоступен после первого сбоя; 3 — стал доступен после первого сбоя и так далее).

Четные числа соответствуют "мертвому"состоянию, нечетные — "живому". Это позволяет избежать отождествления давно работающего и недавно перезагруженного узла. Вектор посещений для каждой Т-функции содержит "0"для узлов, где ни она, ни ее предки не

были, или номер перерождения для тех узлов, на которых данной Т-функции или ее предкам пришлось побывать в соответствующий момент времени.

Т-функция зачастую может быть остановлена, если ее вектор посещений стал неактуален (то есть, один из управляющих предков был утерян). Это значит, что не Т-система следит за разбежавшимися Т-функциями (что даже технически трудноосуществимо) и пытается их остановить, а они сами обнаруживают свою неактуальность и автоматически прекращают свое исполнение.

Если в каком-то вычислительном подпространстве отсутствует (вышел из строя) необходимый ресурс, то Т-функция планируется на исполнение в ближайшее подпространство, где данный ресурс есть в наличии. Возможно, и там не удастся выполнить подзадачу из-за сбоев — тогда последует новый переповтор. То есть, планировщик осуществляет назначение Т-функции на узлы в соответствии с ее и их атрибутами. Также возможно, что будет выведен из строя узел, который активировал объемлющую задачу. Тогда еще более вышестоящий узел сделает переповтор и т д.

7. Достоинства свободной кооперации

Предположим, есть N предприятий (скажем, киностудий), которым нужны суперкомпьютерные мощности. Они совершенно независимы; все что их связывает — это работа в единой сфере. Своя рубашка ближе к телу. Да и суперкомпьютеры "ничьими"не бывают. Понятно, что каждая компания желала бы иметь свой суперкомпьютер, и никак не завязываться на своих "коллег по цеху".

Но после того, как каждая обзаведется своим персональным суперкомпьютером, киностудиям вполне логично кооперироваться. Во-первых, ударный счет им нужен не всегда, и поэтому может образоваться простой мощностей. Во-вторых, суперкомпьютеры очень быстро устаревают, а постоянный апгрейд (что необходимо для следования за технологией) — вещь практически неподъемная. Однако, если N киностудий будут обновлять суперкомпьютеры по очереди, то их общая мощность будет ежегодно подтягиваться за пиком прогресса, ибо суммы N последовательных членов геометрической прогрессии образуют геометрическую прогрессию с тем же коэффициентом роста. При этом каждой киностудии нужно будет приобретать современный суперкомпьютер не ежегодно, а один раз в N лет, что уже гораздо реалистичнее.

Другой пример: химическая промышленность, фармацевтика. Каждому исследовательскому институту хочется иметь свой счетный блок, чтобы не завязываться на других. Но, вполне вероятно, совместно имеет смысл обсчитывать наиболее актуальные, а может быть — даже какие-то общие для всех модели.

Преимущество в обоих случаях в том, что при полной независимости предприятий имеется солидная мощностная и экономическая выгода от кооперации, причем в выборе партнеров предприятия совершенно свободны: кто с кем хочет — тот с тем и кооперируется.

8. Пиковая производительность PlaneT

Где энтузиастам-вычислителям запускать параллельную версию приложения? Можно, конечно, купить небольшой суперкомпьютер. Многим организациям сегодня это уже по карману. А можно, порой, обойтись всем хорошо известным решением — использовать компьютеры, объединенные в локальную сеть. Например, дисплейный класс в институте. Или все дисплейные классы в городе, крае или стране.

Может ли талантливый студент совершить сегодня переворот в науке? Например, доказать или опровергнуть архиважную научную гипотезу? По-видимому, это было возможным всегда, но сегодня имеется еще одна интересная возможность, которой, зачастую, пренебрегают даже ученые. Это численный эксперимент. В науке (например, в математике) много белых пятен, или теорем, к доказательству которых так пока и никто и не смог подступиться. И если написать удачную программу, вычисляющую всего лишь один контрпример, или перечисляющую все классы возможностей, то есть, вероятность попросту "просчитать"неподдающуюся гипотезу. Великая теорема Ферма, теорема о четырех красках, теория солитонов: математики склонны замалчивать роль численного эксперимента, но на самом деле его вклад в в науку трудно переоценить. Один из авторов вспоминает студенческие годы и внезапно возникшую идею о том, как подступиться к опровержению одной из недоказанных теорем. Увы, ЭВМ PDP-11/70 так и не смогла тогда за несколько дней счета дать вразумительного ответа. Но во сколько раз сегодня выше производительность компьютеров! В то же время ясно, что дать постоянный свободный доступ всем школьникам/студентам к современным суперкомпьютерам, которых всегда будет единицы на страну — вещь малореальная. Зато можно объединить всех энтузиастов, и тогда —

чем больше их будет, тем выше будет мощь их совместного метакомпьютера, который всегда будет у них под рукой.

9. Заключение

К сожалению, в настоящий момент кластеры доступны только как автономные вычислительные системы (очередь). К объединению суперкомпьютеров в единое целое пока не все готовы. Будут ли владельцы компьютерных мощностей достаточно лояльны друг к другу, или ограничатся тем, что "своя рубашка ближе к телу"? Станет ли популярным запускать сверхбольшие задачи, или люди предпочтут довольствоваться малым? Вероятно, сегодня на эти вопросы внятного ответа никто не даст. Нашей целью было создание полноценно функционирующего прототипа, на котором можно протестировать несколько демонстрационных задач. Запуски проводились на тех машинах, которые оказались в нашем распоряжении.

Благодарности

Работы, положенные в основу данной статьи, были выполнены в рамках проектов:

- проект "Разработка и реализация языков T++ и соответствующих ему средств для эффективной поддержки высокопроизводительного параллельного счета"по Программе фундаментальных научных исследований ОНИТ РАН "Архитектура, системные решения, программное обеспечение, стандартизация и информационная безопасность информационно-вычислительных комплексов новых поколений"(2009-2010 гг.);
- суперкомпьютерная программа "СКИФ-ГРИД": "Разработка и использование программно-аппаратных средств ГРИД-технологий и перспективных высокопроизводительных (суперкомпьютерных) вычислительных систем семейства "СКИФ"(2007-2010 гг.);
- научно-техническая программа Союзного государства "Развитие и внедрение в государствах-участниках Союзного государства наукоемких компьютерных технологий на базе мультипроцессорных вычислительных систем"(шифр "ТРИАДА") (2005-2008 гг.).

Список литературы

[1] Абрамов С. М., Адамович А. И., Инюхин А. В., Московский А. А., Роганов В. А., Шевчук Ю. В., Шевчук Е. В. *Т-система с открытой архитектурой* // Суперкомпьютерные системы и их применение SSA'2004: Труды Международной научной конференции, 26–28 октября 2004 г., Минск, ОИПИ НАН Беларуси. — Минск, 2004, с. 18–22. ↑1

[2] Абрамов С. М., Есин Г. И., Загоровский И. М., Матвеев Г. А., Роганов В. А. *Принципы организации отказоустойчивых параллельных вычислений для решения вычислительных задач и задач управления в Т-Системе с открытой архитектурой (OpenTS)* // Программные системы: теория и приложения (PSTA-2006): Труды Международной научной конференции, 23–28 октября 2006 г., Переславль-Залесский, ИПС РАН. — Переславль-Залесский : Наука, 2006, с. 257–264. ↑2

[3] Абрамов С. М., Московский А. А., Роганов В. А., Велихов П. Е. Суперкомпьютерные и GRID-технологии. // Пути ученого. Е.П. Велихов / ред.Смирнов В. П.. М. : РНЦ Курчатовский институт, 2007. ISBN 978-5-9900996-1-6. — 314–324 с. ↑4.2

[4] Абрамов С. М., Кузнецов А. А., Роганов В. А. *Кроссплатформенная версия Т-системы с открытой архитектурой* // Параллельные вычислительные технологии (ПаВТ'2007): Труды Международной научной конференции, 29 января –2 февраля 2007 г., Челябинск. — Челябинск : изд. ЮУрГУ, 2007. Т. **1**, с. 115–121. ↑1

[5] Кузнецов А. А., Роганов В. А. *Экспериментальная реализация отказоустойчивой версии системы OpenTS для платформы Windows CCS* // Суперкомпьютерные системы и их применение (SSA'2008): Труды Второй Международной научной конференции, 27–29 октября 2008 г., Минск. — Минск : ОИПИ НАН Беларуси, 2008. — ISBN 978-985-6744-46-7, с. 65–70. ↑1

[6] Есин Г. И., Кузнецов А. А., Роганов В. А. *Экспериментальная реализация отказоустойчивой системы распределенных вычислений "SkyTS"для параллельного счета ресурсоемких T++ приложений в гетерогенной распределенной вычислительной среде* // Программные системы: теория и приложения (PSTA-2009): Труды Международной научной конференции, май 2009 г., Переславль-Залесский/ ИПС им. А.К. Айламазяна РАН ред. Абрамов С. М., Знаменский С. В.. — Переславль-Залесский : изд. Университета города Переславля, 2009. Т. **1**. — ISBN 978-5-901795-16-3, с. 225–244. ↑2

[7] Официальный сайт системы программирования OpenTS : Электронный сетевой ресурс, http://www.opents.net. ↑

A. A. Kuznetsov, V. A. Roganov. *Network topology support in the OpenTS programming system.*

Abstract. Effective aggregation of separate (super)computers requires the use of dynamic parallelization technology, automatic dynamic reconfiguration of computing field, dynamic load balancing, as well as support fault tolerance, taking into account the reliability of nodes and the network topology. The article discusses the domestic software technology developed under the above requirements in the program SKIF-GRID.

Key Words and Phrases: cloud computing, distributed computing, GRID systems, dynamic program paralleling, T-system with an open architecture, fault-tolerance.

Образец ссылки на статью:

А. А. Кузнецов, В. А. Роганов. *Поддержка топологии вычислительного пространства в системе OpenTS* // Программные системы: теория и приложения : электрон. научн. журн. 2010. № 3(3), с. 93–106. URL: http://psta.psiras.ru/read/psta2010_3_93-106.pdf