РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ

АВТОМЕТРИЯ

2006, том 42, № 4

УДК 004.738 : 004.45

ИНТЕГРАЦИЯ ЗАКРЫТЫХ КЛАСТЕРОВ С ИСПОЛЬЗОВАНИЕМ СЕРВИСА МЕЖКЛАСТЕРНЫХ КОММУНИКАЦИЙ *

А. В. Селихов

Институт вычислительной математики и математической геофизики СО РАН, Новосибирск
E-mail: selikhov@ssd.sscc.ru

Представлен новый подход к интеграции вычислительных кластеров для получения большей суммарной производительности. Преодолены ограничения, накладываемые локальностью адресации узлов в объединяемых кластерах. Рассмотрены архитектура Сервиса Межкластерных Коммуникаций, функции его компонентов, а также модель двухуровневого управления приложением на основе Сервиса. Показана зависимость числа коммуникационных каналов между объединяемыми кластерами от количества кластеров и процессов параллельного приложения. Представлены результаты тестирования производительности Сервиса на примере модельной и реальной систем двух закрытых кластеров. Показана эффективность использования Сервиса путем сравнения с модельным случаем использования низкоуровневых коммуникационных функций протокола TCP/IP.

Введение. Растущая потребность в высокопроизводительных вычислениях приводит к необходимости интеграции вычислительных систем на различных уровнях. Разнообразие уровней интеграции, от многоядерных процессоров и SMP-систем до сетей рабочих станций, кластеров, Grid- и P2P-систем, делает возможным выбор наилучшего решения задачи повышения производительности для каждого случая.

Объединение вычислительных мощностей имеющихся кластеров являлось предметом исследования и разработок многих проектов. Несмотря на это, до сих пор существуют некоторые сложности в случае, когда на объединенных кластерах необходимо выполнять параллельные программы, осуществляющие обмен сообщениями между различными процессами. В частности, необходимо использовать дополнительные программные средства для интеграции кластеров, состоящих из узлов с локальными IP-адресами и поэтому недоступных для установления сетевого соединения с любым внешним по отношению к такому кластеру узлом. Имеющиеся реализации интерфейса передачи сообщений (MPI) для Grid-систем, например MPICH-G2 [1]

* Работа выполнена при поддержке Президиума РАН «Программа фундаментальных исследований № 21» (госконтракт 13/4).

или PACX-MPI [2], предполагают существование прямого доступа к любому узлу системы, выполняющему MPI-процесс, что требует наличия у узлов разных кластеров глобальных IP-адресов. Кластеры, в которых только хост-узел доступен извне, можно рассматривать как «закрытые» для любого другого компьютера или кластера. Для такой «закрытости» существует множество причин, включая политику безопасности организации, ограничения на количество используемых глобальных IP-адресов и даже низкий приоритет задачи интеграции кластеров, имеющих высокую исследовательскую или промышленную нагрузку. В таких случаях единственным решением задачи интеграции кластеров является использование хост-узлов для транзита сообщений между вычислительными узлами закрытых кластеров.

Целью представленной работы является решение задачи интеграции вычислительных кластеров в условиях локальной адресации их узлов. Предлагаемый подход к решению задачи основан на концепции Памяти Каналов (ПК) и реализован в виде программного обеспечения среднего уровня. С учетом принципов реализации такой подход назван Сервисом Межкластерных Коммуникаций (СМКК). Он позволяет создать виртуальную разнородную кластерную систему для выполнения параллельных программ с передачей сообщений. Основные функции Сервиса реализуются Шлюзовым Сервером – многопоточной программой, работающей на хост-узлах кластеров и обеспечивающей транзит сообщений, передаваемых между кластерами, а также их буферизацию и конвейеризацию пакетной передачи данных. Использование Шлюзовых Серверов для передачи сообщений становится возможным при помощи другого компонента Сервиса – Клиента, выступающего в роли коммуникационного интерфейса и используемого вместо низкоуровневых функций протокола TCP/IP.

1. Существующие подходы к решению задачи. Сравнение основных существующих решений для обеспечения выполнения параллельных программ с передачей сообщений на Grid-системах приведено в [3]. В соответствии с классификацией, введенной в этой работе, СМКК принадлежит к «шлюзовому» (т. е. основанному на использовании шлюзового сервера) подходу, в котором хост-узел кластера используется в качестве «моста» для объединения внутренней и внешней сетей кластера. Среди наиболее близких по целям и средствам можно выделить проекты PACX-MPI, StaMPI и H2O.

По сравнению с PACX-MPI [2] СМКК не использует дополнительный MPI-процесс, расположенный на одном из узлов кластера, для транзита сообщений между кластерами, а реализует эти функции отдельным многопоточным сервером, запущенным на хост-узле. Это позволяет более целенаправленно использовать вычислительные узлы и хост-узел кластера и, кроме того, не требует наличия внешнего доступа для любого узла, на котором должен быть запущен такой прокси-процесс в PACX-MPI.

В отличие от StaMPI [4] использование Клиент-Интерфейса (КИ) для реализации библиотеки MPI не требует каких-либо дополнительных изменений в коде программы, что может оказаться важным при автоматизации распределения параллельных приложений на кластере, когда в одном случае все приложение работает в рамках одного кластера, а в другом потребуется использование СМКК.

Заслуживающей интерес представляется работа по созданию прокси-модуля в системе H2O [5]. Однако этот прокси-модуль является встроенным в систему и поэтому меньше подходит для использования в других конфигурациях Grid-сервисов в рамках других систем управления приложениями, в то

время как СМКК спроектирован изначально в виде «независимого» сервиса. Кроме того, в системе H2O все коммуникации и внутри кластера, и между кластерами производятся через прокси-модуль, расположенный на хост-узле, что может привести к потере производительности всего параллельного приложения. Наконец, конвейеризованная передача данных между кластерами на основе СМКК показывает более высокую скорость обмена по сравнению с трехэтапной передачей на основе протокола TCP/IP, в то время как в системе H2O наблюдается более низкая скорость по сравнению с такой же трехэтапной передачей.

Среди других средств, используемых для решения одной из поставленных в этой работе задач – обеспечение передачи сообщений между узлами закрытых кластеров, можно отметить два подхода: NAT [6] и VPN [7].

NAT (Network Address Translation) использует средства операционной системы хост-узла для пересылки данных с внутреннего узла на любой внешний узел, имеющий глобальный IP-адрес, который в рассматриваемой задаче может быть адресом хост-узла другого кластера. Недостатком этого подхода является «односторонний» тип устанавливаемого соединения, когда локальный узел одного кластера может посылать все данные только на один глобальный IP-адрес хост-узла другого кластера, являющегося общим (маскирующим) адресом для своих внутренних узлов, в то время как для работы библиотеки MPICH требуется наличие отдельного адреса для каждого узла.

Вопрос уникальности адресов решается в подходе VPN (Virtual Private Network), использующем виртуальные сетевые устройства хост-узлов для передачи данных между локальными узлами кластеров. В этом случае каждый узел всех взаимодействующих кластеров может иметь локальный IP-адрес, однако необходимо обеспечить различие этих адресов в общем адресном пространстве объединяемых кластеров, что, как правило, изначально не соблюдается и поэтому несколько усложняет процедуру объединения кластеров по сравнению с предлагаемым в данной работе подходом. Кроме того, установка VPN-каналов между закрытыми кластерами требует значительно большего объема работ по конфигурированию и настройке хост-узлов кластеров, которая к тому же различается для разных типов UNIX-систем. Наконец, VPN-канал не предоставляет средств буферизации данных и асинхронной передачи, которые могут быть важными для повышения производительности параллельных приложений или для реализации механизмов отказоустойчивости параллельных программ. Тем не менее это не исключает возможности использования VPN для объединения закрытых кластеров.

2. Компоненты Сервиса. Разнообразие задач, которые должны быть решены с использованием Grid-технологии, требует широкого спектра сервисов, ориентированных на решение отдельных подзадач, например сервиса передачи файлов, сервиса авторизации пользователей или сервиса мониторинга состояния вычислительных ресурсов. Для решения различных задач необходимы различные наборы сервисов, и поэтому каждый сервис должен быть настолько независимым («атомарным»), насколько позволяет его целевая функция. В соответствии с этим Сервис Межкластерных Коммуникаций принадлежит к классу программного обеспечения среднего уровня и предоставляет только средства передачи сообщений между закрытыми кластерами, не решая таких важных, но внешних по отношению к этому сервису задач, как выделение множества узлов кластера для параллельной программы, запуск параллельных процессов на узлах в соответствии с их нумерацией в задаче и т. п. Руководствуясь этими принципами, СМКК был спроектирован

как система, состоящая из компонентов двух типов: Шлюза Межкластерных Коммуникаций (шлюзового сервера) и Клиента. Использование СМКК предполагается в рамках высокоуровневой системы управления параллельными приложениями, где задачи распределения параллельных процессов по узлам кластеров и назначения этим процессам идентификаторов приложения и процесса выполняются внешними по отношению к СМКК модулями.

2.1. *Шлюз Межкластерных Коммуникаций.* Как уже было отмечено, единственным средством обмена сообщениями между узлами закрытых кластеров является транзит сообщений через хост-узлы. Для обеспечения асинхронной передачи сообщений через такой разделяемый ресурс Шлюз Межкластерных Коммуникаций (далее Шлюз) должен буферизировать как сообщения, приходящие от Шлюзов других кластеров, так и сообщения, исходящие от узлов своего кластера.

Структура Шлюза представлена на рис. 1.

Наряду с реализацией средства буферизации и формирования очередей сообщений, названного Памятью Каналов [8, 9], Шлюз СМКК осуществляет управление коммуникационными каналами с Клиентами своего кластера и Шлюзами других кластеров. Поскольку общая структура коммуникаций Шлюза является существенно неоднородной, т. е. количество и пропускная способность физических линий связи между узлами одного кластера могут отличаться от аналогичных характеристик коммуникаций между кластерами, организация виртуальных коммуникационных каналов СМКК для асинхронной передачи сообщений представляет отдельный интерес.

Коммуникационные каналы между Шлюзом и Клиентами являются двунаправленными, т. е. передача или прием сообщения может выполняться по одному коммуникационному каналу, поскольку один процесс параллельного приложения не может осуществлять более одной коммуникации в заданный момент времени. Каналы между Шлюзами различных кластеров однонаправлены и разделены на две группы: каналы для сообщений, приходящих локальным процессам от Шлюзов других кластеров, и каналы для сообщений, исходящих от локальных процессов всем другим процессам приложе-

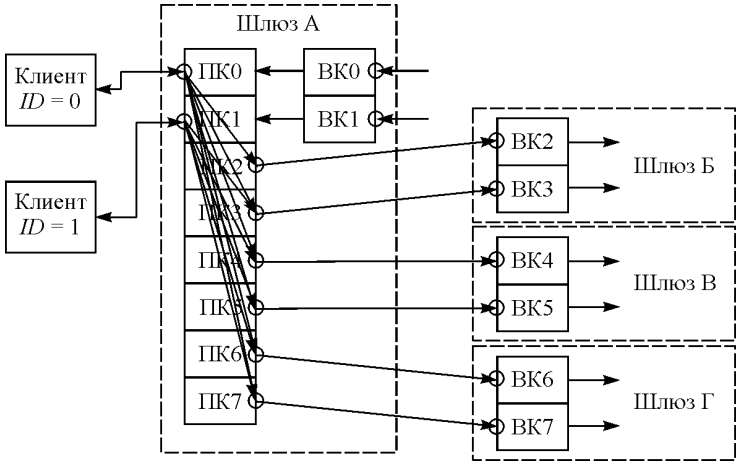

Рис. 1. Структура Шлюза Межкластерных Коммуникаций. ВК – Внешний Канал, хранящий сообщения от других Шлюзов. Стрелками обозначены пути передачи сообщений между Клиентами разных кластеров

ния, контролируемым другими Шлюзами. Сообщения, приходящие от других Шлюзов, буферизуются с целью асинхронной обработки в специальных структурах, названных Внешними Каналами.

Количество коммуникационных каналов L между Шлюзами интегрируемых кластеров, использующих СМКК, является важным параметром пропускной способности и производительности Сервиса. Оно определяется для одного параллельного приложения количеством кластеров (или Шлюзов СМКК) G и общим количеством процессов параллельного приложения N (предполагая без потери общности, что на одном узле кластера выполняется один процесс параллельного приложения). Для рассматриваемой структуры СМКК и Шлюза это количество может быть определено на основе следующего утверждения:

Утверждение. Количество коммуникационных каналов L между G Шлюзами СМКК для параллельной программы, состоящей из N процессов, определяется выражением

$$L = N(G-1). \tag{1}$$

Доказательство. В соответствии с представленной структурой каналов между данным Шлюзом и другими Шлюзами системы, выполняющей одно параллельное приложение, каждый процесс приложения должен использовать каналы Шлюза для получения сообщений от всех других процессов, контролируемых другими $G-1$ Шлюзами. Пусть P_i^{loc} – количество процессов, управляемых i-м Шлюзом, так что $\sum_{i=1}^{G} P_i^{\text{loc}} = N$. Поскольку любой канал исходящих сообщений является каналом входящих сообщений другого Шлюза, общее количество каналов L определяется их суммой для входящих сообщений всех Шлюзов. Для одного Шлюза количество каналов входящих сообщений является произведением количества локальных процессов P_i^{loc} и количества других Шлюзов $G-1$, т. е. $L_i = P_i^{\text{loc}}(G-1)$. Суммирование левой и правой части этого равенства по всем Шлюзам дает искомое выражение (1).

Из выражения (1) следует, что количество каналов между Шлюзами прямо пропорционально количеству кластеров и максимально в случае, если каждый кластер выполняет только один процесс данного параллельного приложения (т. е. количество кластеров равно количеству процессов). Уменьшение количества каналов при уменьшении числа используемых кластеров связано с тем, что часть каналов становится внутренними для кластеров и исключается из числа внешних коммуникационных каналов. Например, для приложения, состоящего из 128 процессов, использование двух кластеров требует наличия 128 коммуникационных каналов между ними, а использование четырех кластеров формирует уже 384 канала.

Выражение (1) определяется структурой Шлюза СМКК и может помочь в принятии решения о количестве используемых кластеров при распределении имеющихся процессов параллельного приложения по узлам кластеров безотносительно к количеству процессов на каждом кластере (что дает свободу действий при распределении вычислительной нагрузки между кластерами).

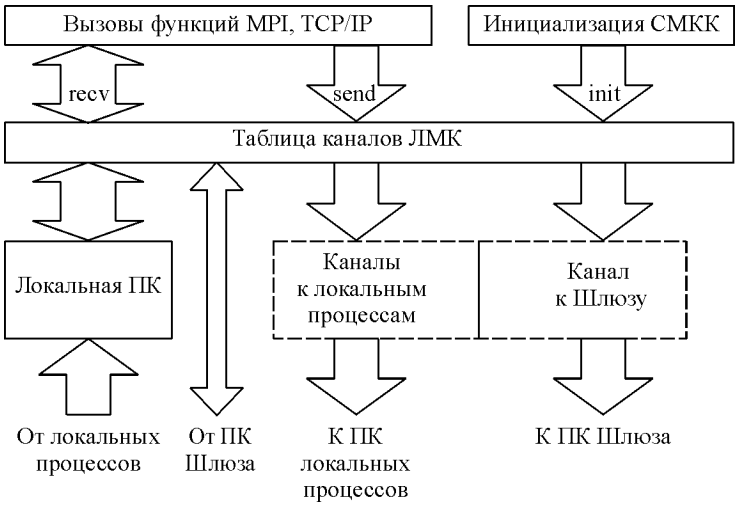

Рис. 2. Структура Клиента СМКК. В Клиент-Интерфейс входят функции передачи (send), приема (recv) сообщений и инициализации (init) клиента. Остальные блоки структуры входят в состав Локального Менеджера Коммуникаций

2.2. *Клиент.* Как было отмечено выше, передача сообщений между узлами закрытых кластеров невозможна с использованием обычных коммуникационных функций протокола TCP/IP и должна осуществляться через хост-узлы, выполняющие функции шлюзов. Для этого необходимо обеспечить процессы параллельного приложения новым средством обмена сообщениями. Таким средством в СМКК является Клиент, состоящий из Клиент-Интерфейса и Локального Менеджера Коммуникаций (ЛМК) (рис. 2).

Клиент-Интерфейс состоит из минимального набора базовых коммуникационных функций приема и передачи сообщений, а также инициализации – завершения работы Клиента. Эти коммуникационные функции являются «обертками» коммуникационных функций протокола TCP/IP. В случае MPI функции КИ заменяют собой функции протокола TCP/IP (такие как send( ) и recv( )) в коммуникациях низкого уровня библиотеки MPICH, реализующих более высокоуровневые пользовательские коммуникационные функции (такие как MPI_Send( ) и MPI_Recv( )). Кроме того, КИ реализует глобальную систему нумерации процессов, использующую номер приложения и номер процесса, что необходимо для обслуживания Сервисом нескольких приложений одновременно.

Локальный Менеджер Коммуникаций, входящий в состав каждого Клиента, связывает идентификатор процесса параллельной программы (например, rank в MPI-программе) с коммуникационными каналами (внешними, через Шлюз, или локальными) с использованием Таблицы каналов ЛМК, а также осуществляет дополнительное взаимодействие со Шлюзом (через канал к Шлюзу (см. рис. 2)) с целью обеспечить асинхронную передачу сообщений. Кроме того, ЛМК управляет локальной ПК, буферизирующей входящие сообщения для данного процесса.

2.3. *Пример конфигурации СМКК.* На рис. 3 представлена конфигурация СМКК для интеграции двух кластеров.

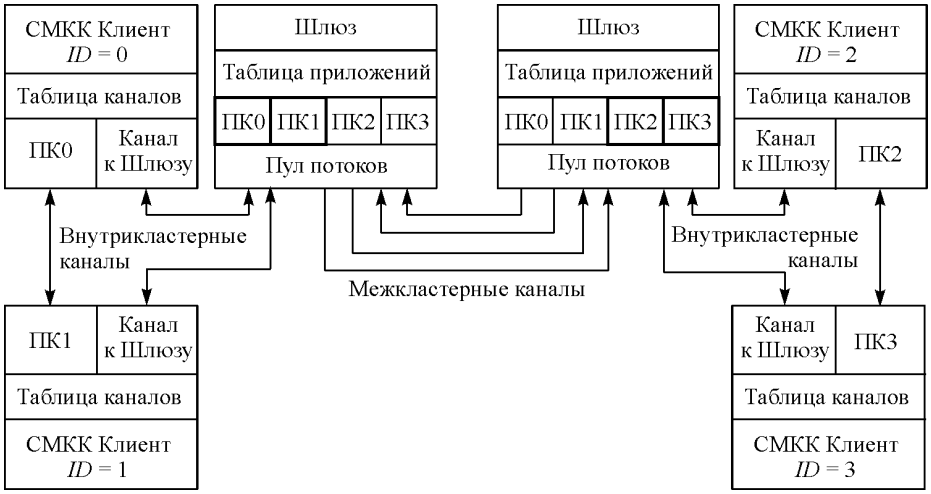

Рис. 3. Пример конфигурации СМКК. Стрелками обозначены коммуникации между элементами и компонентами Сервиса

В этом примере два кластера объединены с помощью СМКК для запуска параллельного приложения, процессы которого выполняются на двух узлах каждого кластера (всего четыре процесса). Клиент обеспечивает коммуникационный интерфейс между процессами приложения и Шлюзом на каждом кластере, в то время как Шлюз осуществляет многопоточную конвейеризированную обработку коммуникаций между процессами разных кластеров. Как видно из рисунка, все коммуникации между процессами приложения и Шлюзами буферизируются с помощью ПК. При этом ПК для локальных коммуникаций внутри кластера организована в рамках Клиента и хранит сообщения, принятые для процесса, обслуживаемого этим Клиентом. Все ПК для межкластерных коммуникаций организованы в структуре Шлюза, что позволяет хранить часть недоставленных сообщений вне процесса-приемника и таким образом разгрузить буферы сообщений процессов параллельного приложения. Каждый запрос на передачу или прием сообщения от Клиента, поступающий на Шлюз, начинает обрабатываться немедленно одним из потоков («легковесных процессов»). Поскольку количество потоков в Шлюзе фиксировано, а количество запросов меняется со временем, в Шлюзе реализован конвейер обработки запросов: каждый поток, занимающийся обработкой запроса от Клиента, передает только часть сообщения, а затем передает запрос в общую очередь и переходит к обработке других (имеющихся или вновь поступивших) запросов от Клиентов. Такая организация работы Шлюза позволяет осуществлять конвейеризацию и асинхронную обработку длинных сообщений, увеличивая тем самым производительность Сервиса. Результаты экспериментов по сравнительной оценке производительности СМКК будут представлены в разд. 4.

3. Модель двухуровневого управления приложениями на основе СМКК. Как было отмечено в разд. 2, СМКК не включает в себя функции по управлению процессами параллельного приложения. Однако с целью иллюстрации способа использования Сервиса в этом разделе приводится модель управления приложением, которая включает в себя СМКК как компонент,

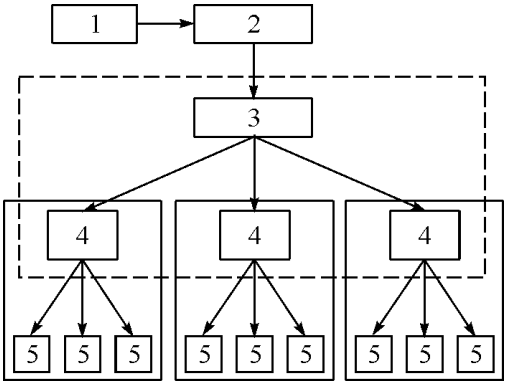

Рис. 4. Схема двухуровневого управления параллельным приложением на основе СМКК: 1 – приложение, 2 – Клиент системы, 3 – Сервер распределения ресурсов, 4 – хост-узлы, на которых запущены Шлюзы СМКК, 5 – узлы, на которых выполняются процессы с использованием Клиентов СМКК

организующий один из уровней общей модели. Схематически модель представлена на рис. 4.

Модель имеет два уровня. На первом уровне закрытые кластеры рассматриваются как мета-узлы в одной из существующих систем управления Grid-ресурсами, например Globus [10], Condor [11] или UNICORE [12]. Приложение поступает на этот уровень управления в виде программы со всеми необходимыми исходными данными, что, как правило, описывается специальной структурой, называемой «паспортом задачи». В таком виде приложение распределяется по хост-узлам кластеров, однако для корректного выполнения параллельного приложения на каждый хост-узел должен поступить также идентификатор приложения и количество его процессов, предназначенных для выполнения на данном кластере. В дальнейшем эта информация используется для инициализации Клиентов СМКК при запуске параллельных процессов. Когда программа вместе с паспортом задачи попадает на хост-узел, она должна быть скомпилирована под архитектуру данного кластера с доступными на кластере библиотеками, в том числе с модифицированной библиотекой коммуникационных функций (например, модифицированной библиотекой MPICH для параллельных приложений, использующих MPI), включающей в себя реализацию КИ.

После успешной компиляции, которая может рассматриваться как этап выполнения задачи на кластере, следующим шагом является запуск процессов параллельного приложения на узлах кластера, при этом распределение процессов по его узлам производится средствами диспетчеризации задач на кластерах. Этот уровень, где процессы параллельного приложения ассоциируются с реальными вычислительными узлами кластеров и производится выполнение реальных вычислительных процессов, а не компиляции, представляет собой второй уровень модели.

Как следует из вышеизложенного, возможности СМКК предоставляются параллельной программе только при ее компиляции с модифицированной библиотекой передачи сообщений, в которую встроены средства поддержки Сервиса. Это может привести, с одной стороны, к потере производительности некоторых параллельных программ на тех кластерах, где есть другие коммуникационные библиотеки (например, «фирменные» реализации биб-

лиотеки MPI, максимально использующие особенности оборудования кластера), обеспечивающие более высокую скорость внутрикластерных обменов. С другой стороны, модификация какой-либо общедоступной библиотеки передачи сообщений, например MPICH, на основе КИ определяет более широкие рамки использования СМКК и его универсальность, что немаловажно при разработке Grid-сервисов. Тем не менее возможность подключения специализированных библиотек для внутрикластерных обменов СМКК является предметом дальнейших исследований.

4. Оценка производительности Сервиса. Производительность СМКК была протестирована на примере реализации библиотеки MPICH с использованием базовых интерфейсных функций КИ. Поскольку основные накладные расходы при работе Сервиса связаны с передачей сообщений между кластерами, эксперименты проводились с целью тестирования именно этой части Сервиса. Производительность коммуникаций внутри кластера полностью зависит от качества программной реализации ЛМК Клиента.

Для оценки влияния архитектуры СМКК на скорость передачи сообщений между кластерами были разработаны параллельные программы, реализующие стандартный тест передачи сообщения от источника к приемнику и обратно (Round-Trip Test). В одной из программ коммуникации были представлены вызовами функций модифицированной библиотеки MPICH, в другой – вызовами функций протокола TCP/IP. Необходимо отметить, что исходный текст MPI-программы содержит коммуникационные функции только библиотеки MPI, т. е. использование СМКК является прозрачным для разработчика параллельных программ.

Поскольку коммуникации между процессами параллельной MPI-программы в этом тесте происходят между узлами различных кластеров, они включают в себя три этапа: две передачи между узлом и Шлюзом и одну передачу между Шлюзами. При этом аналогичный тест на основе функций TCP/IP запускался для каждого из этапов (узел–Шлюз, Шлюз–Шлюз, Шлюз–узел) и фиксировал время коммуникаций каждого этапа, а затем эти три значения времени суммировались и сравнивались с полным временем одного обмена между процессами в MPI-программе.

Тесты проводились на двух различных типах коммуникационных сред. Для тестирования производительности СМКК с минимальным влиянием внешних факторов (например, других потоков данных по коммуникационным каналам и неравномерной загруженности канала) серия экспериментов была проведена в рамках одного кластера, узлы которого связаны через общий коммутатор и коммуникационная нагрузка на кластер была минимальной (первый тип среды). Для проведения эксперимента два Шлюза и два процесса MPI-приложения были запущены на узлах кластера. Эти же узлы использовались для получения значений времени передачи сообщений на основе функций TCP/IP. Результаты эксперимента представлены на рис. 5.

Второй тип коммуникационной среды включал в себя две реальные закрытые вычислительные системы: кластер “mvs1000" Сибирского суперкомпьютерного центра СО РАН и кластер ”mighty" Новосибирского государственного университета. На кластере “mvs1000" постоянно решаются сложные вычислительные задачи под управлением системы очередей, в то время как второй кластер на момент проведения экспериментов был менее загружен. Эти кластеры имеют различную архитектуру процессоров и коммуникационных сред для межпроцессорных обменов, а передача сообщений между хост-узлами проходит через семь промежуточных сетевых узлов. По-

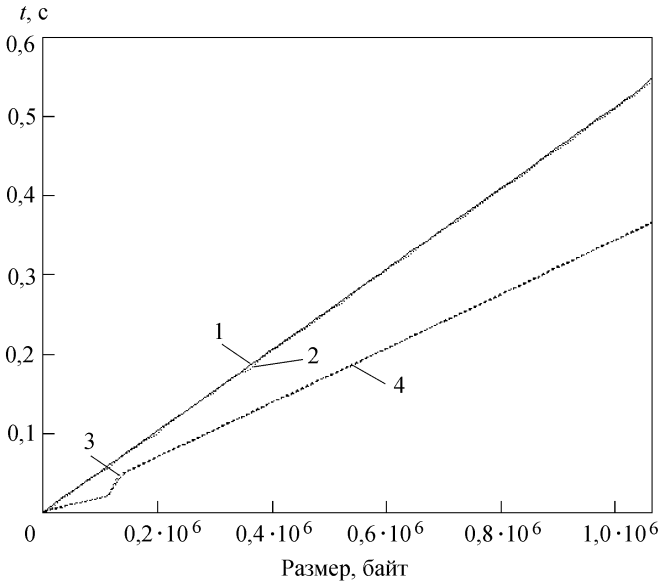

Рис. 5. Результаты тестов на модельной коммуникационной среде с минимальным влиянием внешних факторов: кривая 1 – average, TCP/IP; 2 – min, TCP/IP; 3 – average, ICCS; 4 – min, ICCS (average – среднее значение времени, min – минимальное значение времени на коммуникации, ICCS – значения при использовании СМКК, TCP/IP – суммарные значения при использовании коммуникационных функций протокола TCP/IP)

скольку реальные кластеры, как правило, закрыты для произвольного доступа извне, выполнение тестов потребовало открытие двух дополнительных портов для протокола TCP на каждом хост-узле.

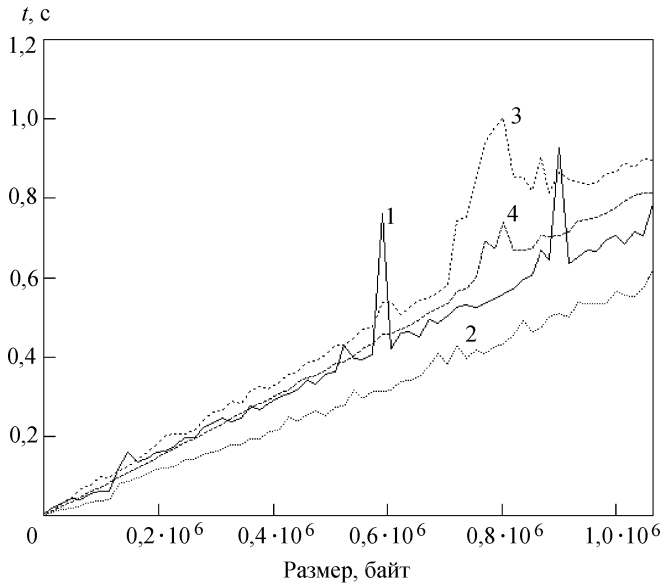

Рис. 6. Результаты тестов на реальной коммуникационной среде, объединяющей два закрытых кластера: кривая 1 – average, ICCS; 2 – min, ICCS; 3 – average, TCP/IP; 4 – min, TCP/IP

Результаты этих экспериментов, представленные на рис. 6, показывают время передачи сообщений через СМКК даже меньшее, чем суммарное время передачи с использованием функций TCP/IP как в модельной коммуникационной среде, так и в реальных условиях. Этот эффект объясняется конвейеризацией пакетной передачи сообщений между узлами, когда часть сообщения начинает передаваться между Шлюзами до того, как все остальные части будут получены Шлюзом.

Заключение. В работе описан Сервис Межкластерных Коммуникаций, позволяющий решить задачу интеграции вычислительных кластеров для запуска параллельного приложения на разных кластерах в случае локализованности адресов их узлов. Предложенное решение основывается на использовании Шлюзовых Серверов на хост-узлах кластеров и Памяти Каналов для буферизации передаваемых сообщений.

Отличительной особенностью предлагаемого решения является его реализация в виде низкоуровневого интерфейса для высокоуровневых коммуникационных библиотек, таких как MPICH, что исключает необходимость модификации параллельной программы для использования Сервиса. Кроме того, для использования СМКК не требуется практически никакой дополнительной настройки операционной системы хост-узлов, а Шлюзовые Серверы являются обычными приложениями и не нуждаются в расширенных правах доступа к системе, что делает использование СМКК безопасным.

Оценка производительности Сервиса показала преимущества использования Памяти Каналов для конвейеризации сообщений, передаваемых между узлами различных кластеров. Количественная оценка на основе проведенных экспериментов иллюстрирует способность СМКК передавать сообщения на 30 % быстрее по сравнению с поэтапной пересылкой на основе функций протокола TCP/IP.

Представленный Сервис позволяет дополнять существующие и разрабатывать новые системы управления выполнением параллельных приложений на Grid, объединяющие кластеры с различной архитектурой и обеспечивающие интеграцию вычислительных мощностей для решения больших задач.

СПИСОК ЛИТЕРАТУРЫ

1. **Karonis N., Toonen B., Foster I.** MPICH-G2: a Grid-enabled implementation of the message passing interface // Journ. Parallel and Distributed Comput. 2003. **63**, N 5. P. 551.
2. **Beisel T., Gabriel E., Resch M.** An extension to MPI for distributed computing on MPPs // Lecture Notes in Comput. Sci. /Eds. M. Bubak, J. Dongarra, J. Wasniewski. Berlin–Heidelberg–New York: Springer-Verlag, 1997. Vol. 1332. P. 75.
3. **Mueller M. S., Hess M., Gabriel E.** Grid enabled MPI solutions for clusters // Proc. of the 3rd IEEE/ACM Intern. Symp. on Cluster Computing and the Grid. Tokyo, 2003. P. 18.
4. **Imamura T., Tsujita Y., Koide H., Takemiya H.** An architecture of stampi: MPI library on a cluster of parallel computers // Lecture Notes in Comput. Sci. /Eds. J. Dongarra, P. Kacsuk, N. Podhorszki. Berlin–Heidelberg–New York: Springer-Verlag, 2000. Vol. 1908. P. 200.
5. **Hwang P., Kurzyniec D., Sunderam V. S.** Heterogeneous parallel computing across multidomain clusters // Lecture Notes in Comput. Sci. /Eds. D. Kranzlmuller, P. Kacsuk, J. Dongarra. Berlin–Heidelberg–New York: Springer-Verlag, 2004. Vol. 3241. P. 337.
6. **http://en.wikipedia.org/wiki/Network_address_translation**
7. **http://en.wikipedia.org/wiki/VPN**

8. **Bosilca G., Bouteiller A., Cappello F. et al.** MPICHV: Toward a scalable fault tolerant MPI for volatile nodes // SC'2002 Conference CD. Baltimore, MD, IEEE/ACM SIGARCH. 2002 (http://www.sc-conference.org/sc2002/paperdfs/pap.pap298.pdf).

9. **Selikhov A., Germain C.** CMDE: a channel memory based dynamic environment for fault-tolerant message passing based on MPICH-V architecture // Lecture Notes in Comput. Sci. /Ed. V. Malyshkin. Berlin–Heidelberg–New York: Springer-Verlag, 2003. Vol. 2763. P. 528.

10. **Foster I., Kesselman C.** Globus: a metacomputing infrastructure toolkit // Intern. Journ. Supercomput. Appl. 1997. **11**, N 2. P. 115.

11. **Litzkow M. J., Livny M., Mutka M. W.** Condor – a hunter of idle workstations // Proc. of the 8th Intern. Conf. of Distributed Computing Systems. 1988. P. 104.

12. **Erwin D. W., Snelling D. F.** UNICORE: a Grid computing environment. Berlin–Heidelberg–New York: Springer-Verlag, 2001. Vol. 2150. P. 825.

Поступила в редакцию 21 ноября 2005 г.