АРХИТЕКТУРА ГЛОБАЛЬНО АДРЕСУЕМОЙ ПАМЯТИ МУЛЬТИТРЕДОВО-ПОТОКОВОГО СУПЕРКОМПЬЮТЕРА

Несмотря на огромные пиковые скорости, эффективность современных суперкомпьютеров для многих задач оказывается чрезвычайно низкой. Дальнейшее масштабирование вычислительных систем все в меньшей мере сказывается на росте их реальной производительности. Выход – в создании новых архитектурных принципов, новых технологий построения как суперкомпьютеров в целом, так и отдельных их подсистем. Именно эту задачу решает отечественный проект создания суперкомпьютера стратегического назначения (СКСН) "Ангара". Статья посвящена одному из важнейших его элементов – системе памяти.

А.Семенов, А.Соколов, Л.Эйсымонт ,к.ф.-м.н.
{semenov, sokolov, verger}@nicevt.ru

Важнейшие проблемы современных высокопроизводительных вычислений – низкая реальная производительность (относительно номинальной пиковой производительности) и низкая продуктивность программирования. Эти проблемы существенно усиливают ряд других серьезных трудностей, таких как высокое энергопотребление, чрезмерная стоимость систем и недопустимая длительность (и цена) разработки приложений. Главные работы по преодолению этих проблем ведут в США фирмы Cray (проект Cascade) и IBM (проект PERCS) в рамках крупной программы DARPA HPCS [1]. Их цель – создание к 2010 году суперкомпьютеров с перспективной архитектурой и реальной производительностью транспетафлопсного уровня (более 10^{15} Flops – PFlops). В России подобные работы выполняются в проекте "Ангара" по созданию суперкомпьютера стратегического назначения (СКСН) с перспективной архитектурой [2, 3].

Главная задача, решаемая в этих проектах, – преодоление так называемой проблемы "стены памяти", т.е. резкого падения реальной производительности из-за простоев процессора, связанных с чрезвычайно неэффективной работой подсистемы памяти. Эта проблема обусловлена, с одной стороны, особенностями элементной базы – задержка обращения к внешней динамической оперативной памяти (DRAM) с учетом промахов в кэш-память дескрипторов сегментов и/или страниц возросла до 300–500 тактов процессора. С другой стороны, в перспективных прикладных программах увеличилась доля команд обращений к памяти, возросла важность невычислительных приложений с интенсивной нерегулярной работой с памятью, все хуже становится пространственно-временная локализация обращений к памяти.

Решение проблемы "стены памяти" в перспективных СКСН предполагает создание подсистемы оперативной памяти (ОП), удовлетворяющей следующим трем требованиям:

- ОП должна быть большой (~10^{15} байт, Пбайт), логически общей, т.е. доступной через единое (глобальное) адресное пространство;
- к ОП необходим эффективный доступ, причем в разных режимах пространственно-временной локализации обращений, что требует особой организации процессора;
- необходимы функционально богатые и эффективные средства взаимодействия и синхронизации процессов, работающих с логически общей памятью.

Большую ОП, доступную через единое адресное пространство, можно составить, только объединив памяти вычислительных узлов мультипроцессорного СКСН. Но для такого объединения необходима межузловая коммуникационная сеть. Задержки обращений к памяти в такой сети составят уже не сотни, а тысячи тактов процессора. Таким образом, проблема обеспечения эффективности доступа значительно усиливается. Кроме того, схема отображения логически единого адресного пространства на распределенную по узлам физическую память должна быть компактной, функционально богатой и гибкой для эффективной реализации параллельных алгоритмов. Это особенно важно при плохой пространственно-временной локализации, так как применить иерархию кэш-памятей в данном случае невозможно.

Наиболее перспективное решение проблемы – обеспечить толерантность процессора к задержкам, т.е. скрыть задержки за множеством одновременных обращений к памяти, выдаваемых с очень высоким темпом. Такой подход используется в векторных процессорах – до 2 тыс. одновременных обращений к памяти в процессоре Cray X1 [4], 8 тыс. – в процессоре Cray BlackWidow [5]. Аналогичный подход реализован и в процессорах с большой мультитре-

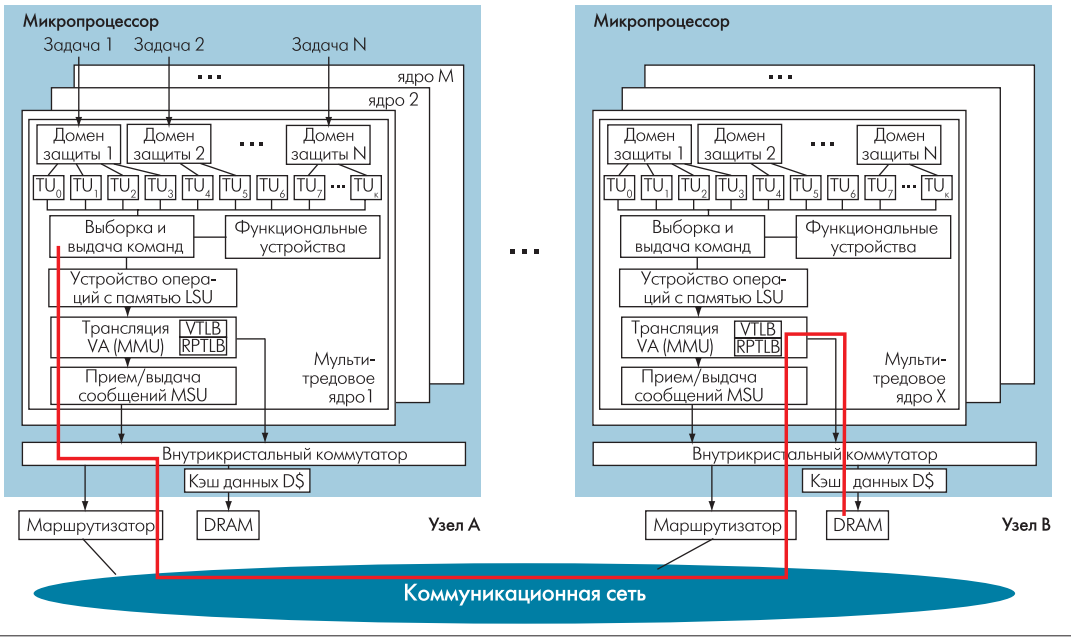

Рис.1. Схема обработки обращения в память удаленного узла

довостью – 1 тыс. обращений от 128 тредов в процессоре Cray XMT [6], а также в ожидаемом мультитредово-векторном сопроцессоре Scorpio перспективной СКСН Cray Cascade [1]. В СКСН "Ангара" выбран такой же подход – до 8 тыс. обращений от 1024 тредов.

Кроме упомянутых трех требований, необходима высокая пропускная способность коммуникационной сети на коротких пакетах, высокий параллелизм выполнения запросов в контроллерах памяти и непосредственно в кристаллах DRAM, а также повышенная надежность работы с памятью [5] – значительно превосходящая ту, что обеспечивается в коммерческих системах с распределенной памятью. Но их рассмотрение – предмет отдельных публикаций.

ОБЩЕЕ ОПИСАНИЕ СКСН "АНГАРА"

СКСН "Ангара" представляет собой вычислительную систему, состоящую из множества вычислительных и сервисных узлов, соединенных коммуникационной сетью. Сеть оптимизирована для передачи коротких пакетов. Рассматриваемый базовый вариант реализации такой сети – 4D- и 5D-торы с адаптивной бездедлоковой* передачей пакетов, возможно модифицированные в соответствии с графами Кэли. Альтернативный вариант – многостадийные сети Клоса [7]. Сервисные узлы базируются на серийных микропроцессорах – зарубежных (Intel, AMD) или отечественных ("Эльбрус 3М", "Багет").

Вычислительный узел (далее – узел) содержит многоядерный мультитредово-потоковый микропроцессор (J7 или J10), локальные модули DRAM-памяти, сетевой маршрутизатор, а также средства обеспечения отказоустойчивости, подключенные к отдельной коммуникационной сети. Каждый узел имеет номер – виртуальный (уровень задачи), а также логический и физический. Логические номера служат для обеспечения отказоустойчивости – при замене отказавшего узла его логический номер присваивается другому узлу.

Микропроцессор содержит до 8 процессорных ядер, каждое из которых включает до 128 тредовых устройств. Задача пользователя может одновременно выполняться в нескольких ядрах микропроцессора. В одном ядре может одновременно выполняться до 16 задач, причем одна из них – операционная система – выполняется всегда. Каждой задаче ядра ставится в соответствие так называемый домен защиты, в котором заданы ресурсы задачи. В архитектурном плане домен защиты – это набор регистров, в которых хранится информация о задаче (таблицы сегментов данных и страниц программ), используемая тредовыми устройствами (TU) при выполнении параллельных процессов – тредов задачи.

TU содержит регистры управления выполнением команд процесса-треда, а также массивы 64-разрядных архитектурных регистров и однобитовых регистров признаков. Тредовые устройства одной задачи совместно используют функциональные устройства ядра [2].

Каждая команда процессора может обращаться к памяти как своего узла, так и удаленных узлов (рис.1). После выборки и выдачи с какого-либо тредового устройства узла A команды обращения к памяти она попадает в функциональный блок LSU подготовки и отслеживания обращений к памяти. Сформированный в LSU исполнительный адрес передается в блок MMU для трансляции виртуального адреса в физический (или в промежуточный глобальный виртуальный адрес). Если адресуемая память прина-

* Дедлок (dead lock) – ситуация, при которой сетевой пакет ждет события, которое не может произойти.

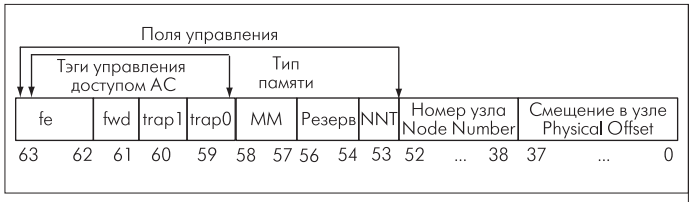

Рис.2. Физический адрес ячейки памяти СКСН "Ангара"

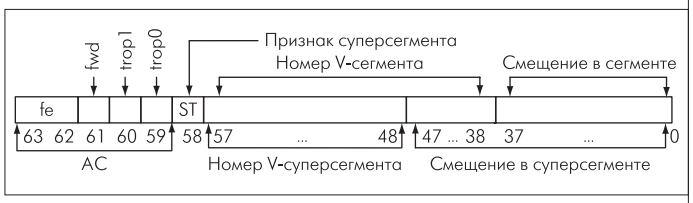

Рис.3. Формат виртуального адреса VA

длежит другому узлу, аппаратно формируется короткий пакет с командой обращения к памяти, который через маршрутизатор направляется в коммуникационную сеть и передается через нее в целевой узел B.

В узле B команда обращения к памяти поступает в блок MMU, где происходит (в случае глобальной виртуальной адресации) дополнительное преобразование глобального виртуального адреса в физический. Далее выполняется обращение к памяти узла, его результат возвращается в узел A в обратном порядке.

ОРГАНИЗАЦИЯ ГЛОБАЛЬНО АДРЕСУЕМОЙ ПАМЯТИ

Адресное пространство узла составляет 256 Гбайт. В системе допускается не более 32768 узлов, соответственно, максимальный объем адресуемой памяти – 8 Пбайт. Вся она входит в единое глобальное адресное пространство. Каждой выполняемой задаче может быть доступно все глобальное адресное пространство (все 8 Пбайт). При этом программы оперируют виртуальными адресами, не связанными непосредственно с конкретными физическими массивами памяти – программа "не знает", на каком именно узле находится ячейка памяти, к которой она обращается. Непосредственная физическая адресация разрешена только в привилегированных режимах ядра ОС и начальной загрузки. Для выбора типа адресации (физическая или виртуальная) служит специальный бит в слове состояния треда. Преобразование виртуальных адресов в физические поддерживается аппаратно.

Физический адрес в СКСН "Ангара" – 64-разрядный (рис.2). Его образуют смещение (физический адрес в локальной памяти узла, 38 бит), номер узла (15 бит) с признаком виртуальной/логической нумерации (бит NNT, 1/0), а также служебные поля управления.

ВИРТУАЛЬНАЯ АДРЕСАЦИЯ

Виртуальная память, доступная задаче, разбита на так называемые V-сегменты. V-сегмент – это область виртуальной памяти, обладающая заданными свойствами и по определенным правилам привязанная (отображенная) к конкретной физической памяти. V-сегментами реализуют защиту данных задачи и распределяют их по физической памяти узлов. Свойства V-сегмента, в частности способ отображения на физическую память, описывается его дескриптором, который хранится в специальной таблице дескрипторов V-сегментов в памяти узла.

Задача оперирует виртуальными адресами VA (рис.3) данных в виртуальном пространстве. VA содержит номер сегмента, смещение в сегменте, а также служебную информацию. Поскольку распределение виртуальной памяти уникально для каждой задачи, для каждой из них в памяти узла формируется своя таблица дескрипторов V-сегментов. В домене защиты задачи хранится физический адрес начала этой таблицы и ее длина. При обращении задачи к памяти происходит выборка дескриптора V-сегмента из таблицы, на основании его информации вычисляется физический адрес или глобальный виртуальный.

Очень важен вопрос быстрой трансляции виртуальных адресов. Для быстрого доступа к дескрипторам V-сегментов предусмотрена специальная кэш-память небольшого размера – VTLB, расположенная в MMU. Кроме того, определены два типа V-сегментов – обычные (от 128 байт до 256 Гбайт) и суперсегменты (от 128 Кбайт до 256 Тбайт). Задача может использовать как до 1 млн. (2^{20}) обычных V-сегментов, так и до 1024 суперсегментов. Суперсегменты введены для работы с очень большими массивами данных как с единым целым, а также чтобы исключить промахи при обращении к VTLB-кэшу. Размеры VTLB-кэша и суперсегментов подобраны так, чтобы через дескрипторы суперсегментов VTLB-кэша можно было адресовать всю доступную в системе физическую память в 8 Пбайт. Аналогичная возможность введена в Cray XMT, но в этой машине объем памяти меньше – 256 Тбайт.

Способ отображения V-сегментов на физическую память задается дескриптором V-сегмента (рис.4). V-сегмент отображается на 2^P узлов со смежными номерами, начиная с узла с

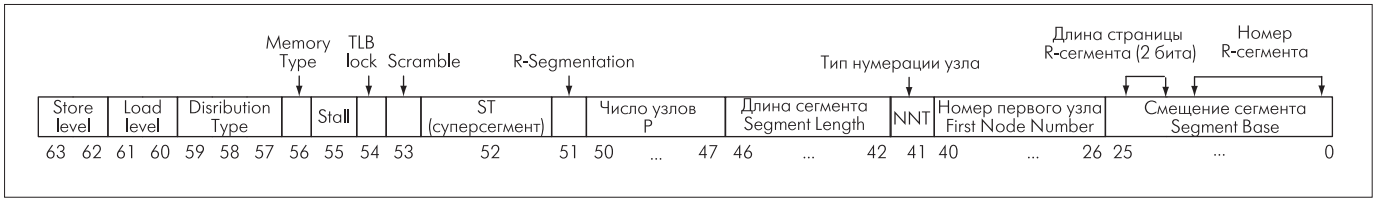

Рис.4. Формат дескриптора V-сегмента

номером First Node Number (эти параметры заданы в дескрипторе). Если используется линейная непрерывная виртуальная нумерация узлов (определяется битом NNT), то соответствующие им логические номера могут быть не последовательными. Однако подобная процедура виртуальной нумерации узлов влечет дополнительные накладные расходы из-за работы с таблицей трансляции виртуальных номеров в логические.

Размер V-сегмента определяется как $S = 128 \cdot 2^{\text{Segment Length}}$ байт для простого сегмента (до 256 Гбайт), для суперсегмента $S = 1024 \cdot 128 \cdot 2^{\text{Segment Length}}$ байт (максимум 256 Тбайт).

Распределение (отображение) V-сегментов на память узлов может быть блочно-циклическим и блочным (рис.5), в зависимости от параметра Distribution Type. В случае блочного распределения Distribution Type = 7, размер блока Z, отведенного в узле для данного сегмента, вычисляется как $Z = S / 2^P$ – в каждом узле только один блок сегмента. При блочно-циклическом распределении поле Distribution Type принимает значение от 0 до 6, сегмент циклически распределяется по узлам блоками размером $B = 4^{\text{Distribution Type}+3}$ (от 64 байт до 256 Кбайт). Причем сначала между $N = 2^P$ узлами распределяются первые N блоков, затем – вторые N и т.д., в одном узле может быть несколько блоков сегмента (см. рис.5).

Отметим, что смещение VA может перед распределением подвергаться дополнительной процедуре – скремблированию, на что указывает бит Scramble дескриптора. При скремблировании смещение обрабатывается по заданному закону псевдослучайного преобразования. Цель процедуры – случайным образом распределить по адресному пространству следующие друг за другом с регулярным шагом виртуальные адреса для снижения вероятности конфликтов при доступе к модулям памяти. При скремблировании младшие 6 бит поля *Смещение в сегменте/суперсегменте* не изменяются.

Размещение данных сегментов непосредственно в памяти узлов (вне зависимости от типа распределения) возможно двумя способами – централизованным и децентрализованным. В первом случае в дескрипторе V-сегмента указан физический адрес области размещения сегмента в памяти узлов, VA-адрес непосредственно транслируется в физический. При децентрализованном размещении точное местоположение сегментов в памяти узлов определяется в самих узлах. В этом случае VA транслируется в так называемый глобальный виртуальный адрес GVA (с точностью до узла), который требует дополнительной обработки в целевом узле. О применении децентрализованного метода указывает бит R-Segmentation дескриптора.

При централизованном способе размещение данных в каждом узле одинаково, первый (или единственный) блок начинается с физического адреса Segment Base, заданного в дескрипторе. Завершается формирование физического адреса переносом в него управляющих полей из VA и дескриптора V-сегмента (AC, NNT).

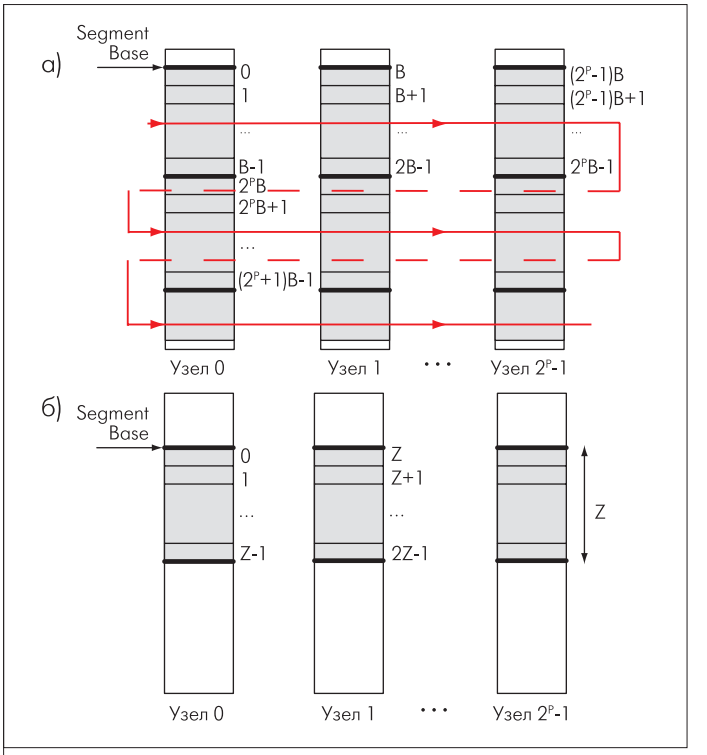

Рис.5. Блочно-циклическое (а) и блочное (б) распределения

Очевидно, что задача централизованного распределения памяти в узлах, с учетом требований защиты данных и возможности одновременного выполнения множества задач, в СКСН весьма сложна [8]. Ведь сегментов может быть больше миллиона, и для каждой задачи их распределение уникально – но при этом не должны возникать конфликты обращения к физической памяти. Для упрощения этой проблемы в СКСН "Ангара" введен способ децентрализованного управления памятью узлов на основе R-сегментов.

При децентрализованном способе трансляции VA формируется не физический, а глобальный виртуальный адрес GVA (рис.6). При трансляции виртуального адреса по уже рассмотренной процедуре вычисляется номер узла (Node Number), определяется смещение во фрагменте сегмента и записывается в поле *Смещение во фрагменте R-сегмента*. Дополнительно из дескриптора копируются номер R-сегмента и длина страницы R-сегмента. В результате получается 87-разрядный адрес, который передается в целевой узел Node Number (если, конечно, адресуемая память не принадлежит узлу-отправителю). Дальнейшее преобразование GVA уже в физический адрес происходит в блоке MMU целевого узла.

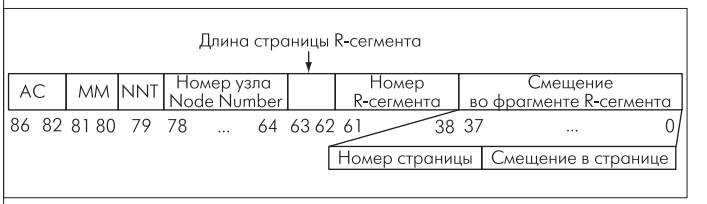

Рис.6. Формат глобального виртуального адреса GVA

Глобальная виртуальная адресация использует понятие R-сегментов. В отличие от V-сегментов, номера R-сегментов общие для всей системы. В каждом узле фрагмент R-сегмента отображается на физическую память страницами размером по 16 Кбайт, 256 Кбайт, 4 Мбайт или 64 Мбайт. Размер страниц указан в дескрипторе V-сегмента и при трансляции переносится в GVA. GVA содержит достаточно информации для формирования физического адреса в целевом узле – остается только определить начальный адрес страницы. Именно это и происходит при трансляции GVA в блоке MMU узла.

Идея R-сегментов используется также в системах Cray T3E и Cray X1. Переменный размер страниц поддерживается в современных микропроцессорах. Например, в процессоре Barcelona (AMD) поддерживаются страницы размером 4 Кбайта и 2 Мбайта. Самый большой размер страницы в системе Cray BlackWidow – до 4 Гбайт.

ТРАНСЛЯЦИЯ ВИРТУАЛЬНЫХ АДРЕСОВ

Трансляция виртуального адреса в физический или глобальный виртуальный адрес начинается с выборки дескриптора V-сегмента (рис.7), соответствующего данному VA. Сначала из адреса считывается номер сегмента (см. рис.3), причем из разных разрядов в зависимости от типа (обычный или суперсегмент), что определяется битом ST. Номер сегмента однозначно указывает на его дескриптор – либо в кэш-памяти VTLB, либо в таблице дескрипторов (рис.7). Таблица дескрипторов в памяти узла состоит из двух частей – для обычных и суперсегментов. В регистре домена защиты указан ее начальный адрес и длина. В таблице смещение адреса дескриптора обычного сегмента соответствует номеру сегмента. Смещение адреса дескриптора суперсегмента вычисляется как *Длина таблицы* минус *Номер суперсегмента*.

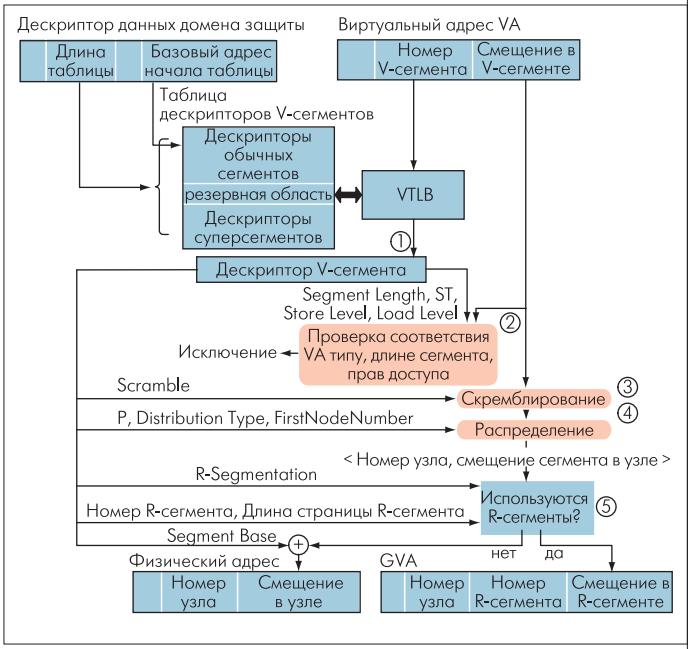

Рис.7. Схема трансляции виртуального адреса

Отметим, что помимо VTLB, дескрипторы кэшируются и в кэш-памяти данных микропроцессора D$, поэтому при промахе в VTLB происходит обращение в кэш данных и лишь затем – в память узла. Кэш-память VTLB является общей для всех доменов защиты ядра микропроцессора, поэтому дескриптор V-сегмента попадает в нее вместе с номером его домена защиты.

После выборки дескриптора V-сегмента начинается аппаратная проверка соответствия свойств виртуального адреса и V-сегмента: сопоставляются их поля ST, контролируется выход VA за границы сегмента (поля *Смещение в сегменте/суперсегменте* сравниваются с длиной сегмента). Кроме того, заданный в дескрипторе минимально необходимый уровень доступа к сегменту по записи и чтению (Store level и Load level) должен соответствовать уровню привилегий из слова состояния треда, выдавшего обращение. Если при проверке обнаружены нарушения, то возбуждаются соответствующие исключительные ситуации и трансляция адреса аварийно завершается.

В случае успешности проверки вычисляется адрес – физический или GVA. Трансляция происходит в блоке MMU каждого ядра микропроцессора (см. рис.1). В специальных регистрах MMU хранит не только логический номер узла, но и виртуальный номер узла в рамках решаемой задачи. Это позволяет распознать обращения к внутриузловой памяти. Обращения к другим узлам будут передаваться в блок MSU приема/выдачи сообщений (см. рис.1), в котором имеется таблица привязки виртуальных номеров узлов к их логическим номерам, которая используется в случае обращения по виртуальному номеру узла.

Кэш-память VTLB имеет наборно-ассоциативную организацию, содержит 4 набора по 512 записей (всего 2048 дескрипторов). Для поиска дескриптора в VTLB используется ключ, составленный из его смещения в таблице дескрипторов V-сегментов и номера домена защиты.

Дескриптор можно сделать резидентным в VTLB, для этого служит поле TLB-lock. Такой дескриптор не будет выталкиваться из VTLB, пока его явно не удалят. Если необходимо динамически изменить информацию в дескрипторе V-сегмента, работу с ним нужно временно заблокировать. Для этого служит бит Stall. Команды, обращающиеся к дескриптору с установленным битом Stall, будут переповторяться, но не выполняться, пока бит не будет сброшен.

Предусмотрены команды непосредственной работы с VTLB. Например, при помощи команды VTLB_FLUSH_EXT при выгрузке задачи можно очистить VTLB от всех ее дескрипторов. Другой командой можно удалять из VTLB отдельные дескрипторы по номеру V-сегментов.

УРОВЕНЬ R-СЕГМЕНТОВ – РАЗМЕЩЕНИЕ ДАННЫХ ВНУТРИ УЗЛОВ

Трансляция GVA в физический адрес (рис.8) в узле сводится к поиску адреса начала заданной страницы. Он хранится в дескрипторе страницы (рис.9). Дескрипторы страниц образуют

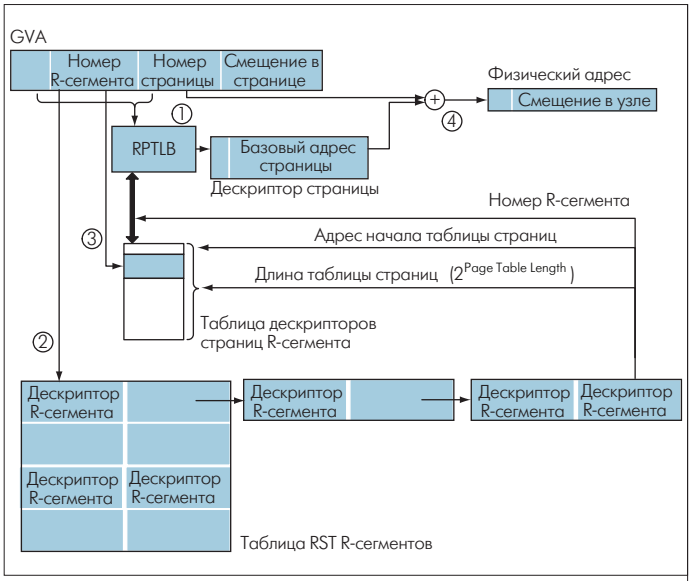

Рис.8. Трансляция глобального виртуального адреса в физический адрес памяти узла

таблицу дескрипторов страниц в памяти узла. Кроме того, дескрипторы страниц могут попадать в специальную кэш-память RPTLB. RPTLB – наборно-ассоциативная кэш-память (4 набора по 256 записей). В каждой записи хранятся номера R-сегмента и страницы, а также физический адрес страницы. Для поиска в RPTLB используется ключ, составленный из номера R-сегмента, номера страницы и ее длины.

Если дескриптор страницы обнаружен в RPTLB, задача трансляции фактически решена. При промахе в RPTLB необходимо обратиться к таблице дескрипторов страниц R-сегмента. Ее адрес (старшие 26 разрядов адреса) указан в дескрипторе R-сегмента (рис.10). Каждый R-сегмент имеет уникальный номер в пределах всей системы, но в узле, как правило, используется ограниченное число R-сегментов. Для хранения их дескрипторов используется хэшированная таблица дескрипторов фрагментов R-сегментов RST (Real Segment Table). Ключом поиска дескрипторов в таблице RST служит номер R-сегмента. Одному значению хэш-функции (от номера R-сегмента) соответствует одна строка таблицы. В одной строке хранится либо два дескриптора (если данному значению хэш-функции соответствует не более двух номеров R-сегментов), либо дескриптор и ссылка на следующую пару (два дескриптора или дескриптор/указатель). Наличие значимой информации показывают биты Valid-0 и Valid-1 (если они равны 1). Бит Descriptor указывает, что находится во втором элементе – дескриптор (Descriptor = 1) или ссылка.

Если дескриптор нужного R-сегмента найден, блок MMU проверяет совпадение размера страницы в GVA и в дескрипторе R-сегмента. Кроме того, контролируется, что номер страницы из GVA не превышает размера таблицы страниц данного R-сегмента ($2^{\text{Page Table Length}}$). При обнаружении какой-либо ошибки возбуждается соответствующая исключительная ситуация. Если ошибок нет, то из найденного дескриптора R-сегмента выбирается *Адрес начала таблицы страниц* (старшие 26 разрядов адреса), к нему прибавляется номер страницы, и по полученному адресу происходит обращение к таблице дескрипторов страниц. В дескрипторе страницы указаны старшие 26 разрядов ее физического адреса в памяти узла. При обращении к дескриптору страницы он вместе с номерами R-сегмента и страницы автоматически заносится в RPTLB.

Полученный физический адрес начала страницы складывается со смещением внутри страницы, выбранным из GVA. На этом формирование физического адреса завершено.

БАЗОВЫЕ АДРЕСУЕМЫЕ ЭЛЕМЕНТЫ ПАМЯТИ И ОПЕРАЦИИ С НИМИ

Физическая и виртуальная память данных адресуется до байта, однако основным рабочим объектом памяти является 64-разрядное слово – ячейка. Для эффективной синхронизации* при работе с памятью предусмотрен ряд средств, использующих теги доступа и состояния ячейки (внешние и внутренние). Каждой ячейке (рис.11) поставлены в соответствие два внешних теговых бита состояния ячейки. Один из них – признак заполненной/пустой ячейки (full/empty-бит, fe-бит). Другой тег состояния (extag-бит) показывает, используются ли внутренние теги (три младших бита ячейки). Если ячейка содержит данные, можно использовать только внешний fe-бит. Внутренние теговые биты состояния могут присутствовать только в ячейках, содержащих адрес, поскольку для адресации 64-битовых ячеек младшие три бита адреса не нужны. Назначение внутренних теговых битов: fwd-бит – признак косвенной адресации, биты trap0 и trap1 – признаки необходимости аппаратного возбуждения соответствующих исключительных ситуаций при доступе к ячейке.

Для работы с тегами состояния в адрес (физический и виртуальный) введены теги управления доступом – старшие

Рис.9. Дескриптор страницы R-сегмента – элемент таблицы страниц

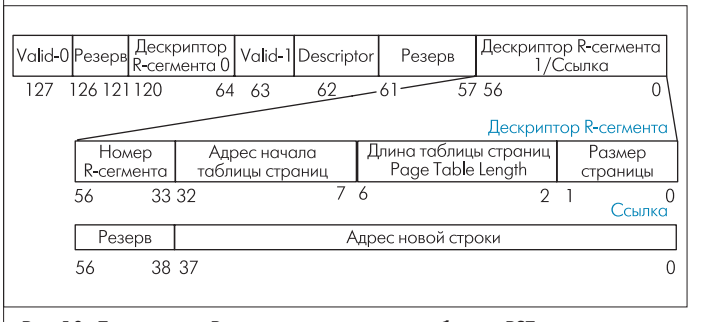

Рис.10. Дескриптор R-сегмента – элемент таблицы RST

* Синхронизация – механизм обеспечения правильного взаимодействия параллельных процессов.

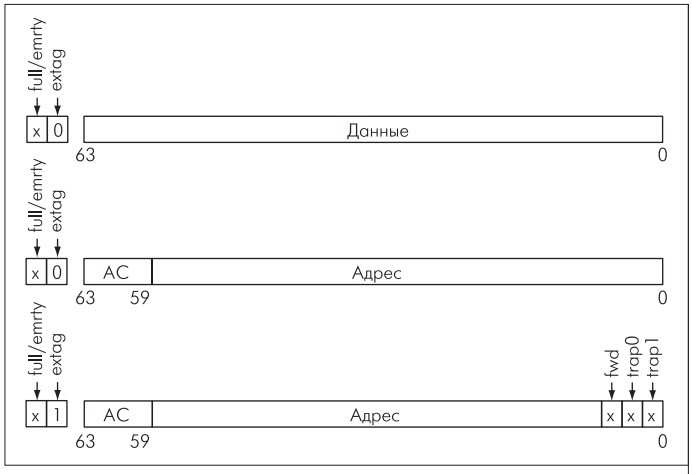

Рис.11. Базовые адресуемые элементы памяти с теговыми битами состояния

пять бит адреса, поле AC (Access Control). Биты этого поля показывают, используются ли внешние/внутренние теги и в каком режиме. Первые два бита поля AC – fe_control-биты – задают режим работы:

00 – *Normal*, обычный режим работы с ячейкой памяти без учета теговых битов;

10 – *Future*, операция выполняется, только если fe-бит в адресуемой ячейке равен 1 (full). После выполнения операции у ячейки остается состояние full. Режим позволяет заблокировать доступ к ячейке с состоянием empty, что важно для обеспечения правильного взаимодействия параллельных процессов (например, для барьерной синхронизации);

11 – *Synchronize*. В этом режиме по чтению можно обращаться только к ячейкам в состоянии full, по записи – к ячейкам в состоянии empty. После выполнения операции состояние инвертируется. В таком режиме возможно выполнение только последовательности команд "запись – чтение – запись – чтение ..." (нельзя дважды подряд считать или записать данные). Это удобно для реализации "почтовых ящиков" между процессами.

Остальные три бита AC соотносятся с внутренними тегами – запрещают или разрешают их использование. Биты поля AC устанавливаются в программе пользователя, причем в системе команд для этого предусмотрено несколько механизмов.

Использование теговых битов состояния и доступа достаточно разнообразны. В системе команд имеются операции явной работы с теговыми битами. Механизм тегов – это гибкое и эффективное средство обеспечения низкоуровневой синхронизации процессов при работе с памятью. Аналогичный механизм уже хорошо зарекомендовал себя в таких системах, как Cray MTA-2 и Cray XMT [6, 9].

Для примера рассмотрим операции доступа к адресуемой ячейке памяти в режиме Synchronize.

Операция чтения выполняется только для состояния full адресуемой ячейки памяти, после выполнения операции состояние ячейки меняется на empty. Если состояние адресуемой ячейки empty, то микропроцессор переходит в режим "ожидание" – начинает повторять команду чтения, пока не изменится состояние ячейки. Для каждой задачи в домене защиты можно задать максимальное число повторов выполнения команды. При его превышении возбуждается соответствующая исключительная ситуация. При этом (в стандартном случае) в адресуемой ячейке устанавливается trap-бит, а также формируется привязанный к ячейке список тредов, которые пытаются читать содержимое данной ячейки. Активация тредов списка произойдет только после записи значения в эту ячейку. Факт наличия списка тредов распознается по установленному trap-биту.

Операция записи в режиме Synchronize выполняется только для адресуемой ячейки в состоянии empty, по ее выполнению состояние меняется на full, если к ней не был привязан список тредов-читателей. Если такой список был, то первый тред-читатель выбирается из списка и получает значение, ячейка остается в состоянии empty. Если же состояние этой ячейки full, то также происходит ожидание, которое может закончиться записью или установкой trap-бита с организацией привязанного к ячейке списка тредов, которые пытаются записать данные в ячейку.

Отметим, что аппарат управляемого переповтора команд доступа к памяти обеспечивает богатые возможности взаимодействия и синхронизации при работе с глобально адресуемой памятью множества.

Помимо механизма тегов, для синхронизации тредовых процессов в СКСН "Ангара" введены атомарные операции, которые на уровне команды блокируют доступ к ячейке на время выполнения с ней операций. Например, команда Acswp вида (Acswp <адрес><операнд 1><операнд 2>) считывает значение из ячейки памяти по заданному адресу и блокирует к ней доступ. Затем считанное значение сравнивается с содержимым операнда 1. Если они совпадают, то значение операнда 2 записывается в ячейку памяти, в противном случае записи нет. По завершении команды снимается блокировка доступа к ячейке. Команда Acswp возвращает старое значение ячейки.

ОЦЕНКА ЭФФЕКТИВНОСТИ ОРГАНИЗАЦИИ ПАМЯТИ СКСН "АНГАРА"

Оценка эффективности организации памяти СКСН "Ангара" сводится к двум вопросам:

- эффективность реализации виртуальной памяти. Критерий оценки: эффективность выполнения задач с виртуальной адресацией должна быть близка к достигаемой при физической адресации;
- степень обеспечения богатыми и гибкими средствами, достаточными для разработки высокоэффективных и компактных параллельных программ над общей памятью огромного объема. Эти программы должны превос-

Таблица 1. Длительность различных обращений к памяти для микропроцессора J7 (500 МГц) и памяти DDR2 (800 МГц). Задержки приведены для системы без нагрузки, они являются минимальными, для загруженной подсистемы памяти и сети они значительно вырастут

Вид обращения	Время, тактов процессора
По физическому адресу в кэш данных	19
По физическому адресу в память (промах в кэш данных)	52
По виртуальному адресу в кэш данных	32
По виртуальному адресу в память	68
По виртуальному адресу в память с промахом в VTLB и RPTLB	249
По виртуальному адресу в память соседнего узла, расстояние 1	425
По виртуальному адресу в память соседнего узла с промахами в VTLB и RPTLB	606
По виртуальному адресу в память узла на расстоянии 7	790
По виртуальному адресу в память узла на расстоянии 7 с промахами в VTLB и RPTLB	971

ходить уровень, достигнутый на существующих суперкомпьютерах при использовании стандартных языков программирования и библиотеки MPI.

Для исследований СКСН "Ангара" была разработана параллельная потактовая имитационная модель (на языке Charm++), хорошо масштабируемая по производительности. Накладные расходы при работе с глобально адресуемой памятью вносят существенные задержки выполнения операций (табл.1). Причем механизмы трансляции виртуальных адресов должны в большей степени сказаться на эффективности выполнения задач в одном узле, поскольку при работе с сетью определяющим фактором будет уже коммуникационная задержка сети и пропускная способность сетевого интерфейса. Поэтому отдельно рассмотрим производительность одного узла и множества узлов.

Производительность одного узла

В исследованиях на одном узле рассматривалась производительность четырех задач с разной пространственно-временной локализацией – dgemm, БПФ, STREAM Triad, RandomAccess. При этом изучалась зависимость производительности от числа тредов, а также от использования адресации по физическим или виртуальным адресам. Для однотредовых программ производительность при виртуальной адресации оказалась примерно на 20% ниже, чем непосредственно при работе по физическим адресам, что является неплохим показателем. Это говорит о том, что число обращений к памяти даже при одном треде достаточно велико, чтобы проявилось свойство толерантности к задержкам операций с памятью.

Отметим, что часть обращений вызывали промахи в TLB-кэши. Толерантность процессора к задержкам при промахах TLB-кэши – уникальное явление, поскольку организация этих TLB-кэшей с целью минимизации промахов всегда считалась очень трудным делом. При большом числе тредов производительность при работе по физическим и по виртуальным адресам вообще выравнивается. Следовательно, толерантность позволяет скрывать накладные расходы на реализацию виртуальной памяти даже на уровне одного узла.

Производительность должна зависеть от числа параллельных каналов блока MMU (число адресов, принимаемых на трансляцию за один такт). Оказалось, что достаточно принимать в MMU два адреса за такт, а для процессора J7 – один адрес за такт.

Тесты на множестве узлов

При работе со множеством узлов добавляются существенные задержки сети. Для оценки производительности операций чтения/записи в память удаленного узла использовался стандартный PUT/GET-тест [10] для линка с пиковой пропускной способностью 12 и 4 Гбайт/с (рис.12). Видно, что пропускная способность зависит от числа тредов, причем на больших длинах пакетов ее уровень достаточно высок и приближается к пиковой пропускной способности канала связи. Насколько хороши полученные результаты, позволяет судить еще один эксперимент на 4-Гбайт/с линках (уровень канала Infiniband 4xQDR). В этом случае продемонстрирована реальная пропускная способность, не достижимая в системах на базе коммерчески доступных компонентов даже при аппаратном копировании данных через интерфейс прямого доступа к памяти. В СКСН "Ангара" пересылка организована программно, хоть и с использованием 64 тредов.

Результаты исследования эффективности на синтетическом тесте APEX-MAP [11] показывают резкое улучшение характеристик доступа к памяти в СКСН "Ангара" для режимов плохой пространственно-временной локализации [2, 3]. APEX-поверхность имеет вид практически горизонтальной плоскости, а не "горки", характерной для существующих суперкомпьютеров. Этой поверхности в це-

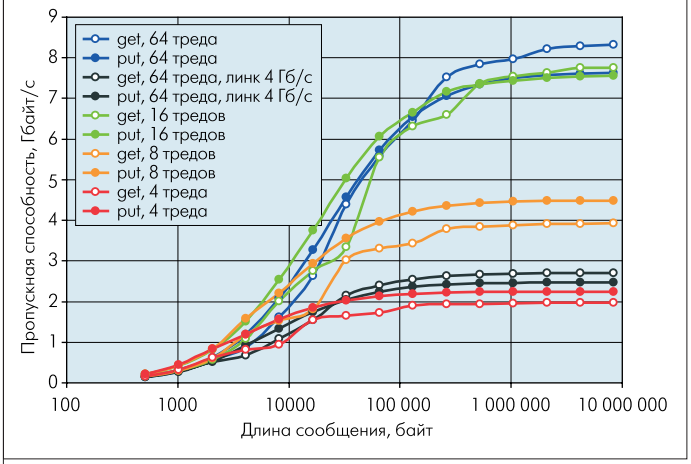

Рис.12. Пропускная способность при выполнении операций put и get с соседним узлом для разного количества тредов, процессор J7-2, два узла, одно ядро, векторные операции по 8 слов

Таблица 2. Сравнение СКСН на тестах с разной локализацией обращений к памяти

Суперкомпьютер	BlueGene/P	BlueGene/L	Cray XT5	Cray X1E	"Ангара", J7-2	"Ангара", J10-4
Число узлов	32768	65536	74529	1008	1024	1024
RandomAccess, GUPS	103,1800	35,4706	16,6115	7,6881	65,7	175,32
Умножение матриц, тест HPL, TFlops	173,4	259,2	901,9	14,9	3,1*	103,9*
БПФ, GFlops	5080	2311	2773	245	605	–
BFS, ME/s	–	7655 (32768 узлов)	–	–	21021	55635

* Тест MMult.

лом соответствует хороший уровень эффективности. Таким образом, тест APEX-MAP демонстрирует достижение основной цели разработки СКСН "Ангара" – эффективной работы с глобально адресуемой памятью большого объема с приблизительно равными показателями в разных режимах пространственно-временной локализации. Отметим, что тест APEX-MAP работал с виртуальными адресами.

Помимо тестов APEX-MAP, реальная производительность системы оценивалась и на задачах пакета HPC Challenge [12] (табл.2). Эта методика использует тесты с разной пространственно-временной локализацией обращений к памяти. Тесты умножения матриц (MMult и HPL), одномерного быстрого преобразования Фурье (БПФ), STREAM Triad и RandomAccess соответствуют границам области реальных прикладных задач по пространственно-временной локализации их обращений к памяти.

Лучший практический результат на тесте RandomAccess на ноябрь 2008 года – 103,9 GUPS. В программе DARPA HPCS [1] поставлена цель достижения на перспективных американских СКСН 64000 GUPS. На модели СКСН "Ангара" достигнуты показатели, значительно превышающие существующие значения. Так, в максимальной конфигурации – 32768 узлов с микропроцессором J10-4 – получено значение 2700 GUPS. Оценка резервов оптимизации коммуникационной сети с топологией 4D-тора позволяет надеяться на достижение 4000–5000 GUPS. Дальнейшее улучшение возможно только при использовании топологии тора более высокого порядка или сети Клоса [7].

Тесты умножения матриц MMult [13] и HPL имеют близкие показатели хорошей пространственно-временной локализации, поэтому их результаты можно сравнивать между собой. Результаты сравнения тестов MMult и HPL важны тем, что современные суперкомпьютеры развивают на них высокую реальную производительность благодаря виртуозно написанным программам с эффективным использованием иерархии кэш-памяти. В СКСН "Ангара" нет мощной иерархии кэш-памяти. Тем не менее достаточно простая мультитредовая программа MMult для этого компьютера показала фактически аналогичные результаты по эффективности (см. табл.2).

Для оценки одномерного БПФ использовался мультитредовый алгоритм Npoints [14]. Это тест с плохой пространственной локализацией. Реальная производительность СКСН "Ангара" в пересчете на один узел на порядок превысила результаты других современных суперкомпьютеров.

Поскольку СКСН "Ангара" предназначен в первую очередь для эффективного решения задач с интенсивной нерегулярной работой с памятью, дополнительно использовался наиболее известный оценочный тест такого типа – BFS (Breadth-First Search, поиск вширь в графе) [15]. Характеристика производительности этого теста – число миллионов ребер графа, пройденных за секунду (ME/s). Для BlueGene/L обработан наибольший граф из всех известных на ноябрь 2008 года, состоящий из $3{,}2 \cdot 10^9$ вершин. На этом СКСН достигнута наивысшая известная производительность данного теста на реальном оборудовании, однако при этом используется очень много (32768) узлов. СКСН "Ангара" существенно превосходит этот результат при значительно меньшем числе узлов (табл.2).

Эффективность средств работы с памятью

Тесты СКСН "Ангара" выполнялись при помощи программ, написанных на языке ассемблера с использованием глобально адресуемой памяти. Однако тесты почти всех первых вариантов программ были неудачными. И лишь при использовании всего многообразия и гибкости средств работы с глобально адресуемой памятью и мультитредовости СКСН "Ангара" были достигнуты высокие результаты. Рассмотрим особенности реализации тестовых программ.

В тесте RandomAccess изменяемый массив расположен в виртуальном сегменте с блочно-циклическим распределением и скреблированием. Это отображение оптимально, поскольку тест обладает наихудшей локализацией обращений к памяти.

Для тестов БПФ и умножения матриц применяется программная гетерогенная тредовая модель – используется принцип разделения работы по доступу к данным и вычислениям. Часть тредов узла занимается подкачкой данных в его память из глобальной памяти и записью полученных данных обратно в глобальную память. Другие треды выполняют вычисления над данными, локализованными в памяти узла. Таким образом, треды делятся на две части – вычислительные треды и треды работы с данными. Кроме того, при отображении виртуальных сегментов использовалось скреблирование. Еще большая оптимизация производительности возможна за счет блочного распределения этих сегментов и соответствующего разделения работы между узлами.

Высокие результаты теста BFS достигнуты благодаря программам, использующим команды вызова процедур на удаленных узлах (RPC) и одноименный алгоритм [16] с неоднородными тредовыми вычислениями. Этот алгоритм особенно эффектно использует возможности управления локализацией в глобально адресуемой памяти. Для самого большого массива, в котором хранятся все списки смежных вершин графа, используется сегмент с блочным распределением, с соответствующим распределением функций между узлами, чтобы обращения в этот массив были по возможности локальными. Однако из-за неоднородности графа часть списка смежных вершин все-таки оказывается в памяти соседних узлов, но глобально адресуемая память делает доступ к ним прозрачным для программы. Кроме массива, распределенного на множество узлов, в алгоритме используются и локальные массивы, полностью отображенные на память конкретных узлов (с возможностью внешней адресации). За счет разнообразного управления локализацией достигается простота и эффективность алгоритма RPC, что и обеспечивает рекордные результаты.

Таким образом, проведенные исследования доказали возможность достижения реальной производительности, заявленной в проекте СКСН "Ангара".

Внедрение глобально адресуемой памяти может повлечь разработку новых алгоритмов и позволить решать такие задачи, о которых раньше и не мечтали. Это явно проявилось в процессе внедрения Cray XMT [17]. Появляющаяся техника такого типа настолько необычна, что одновременно идет подготовка сообщества прикладных программистов из области высокопроизводительных вычислений высшего уровня к появлению перспективных СКСН Cray Baker и последующих СКСН программы DARPA HPCS [1]. Аналогично, в проекте СКСН "Ангара" этапы предварительных НИР завершены, наступило время оценки системы уже не на тестах, а на ядрах и фрагментах существующих и перспективных приложений. Поэтому необходимо взаимодействие с широким кругом пользователей. В дальнейших публикациях мы расскажем о других принципиальных особенностях архитектуры и системах СКСН "Ангара" и приемах их эффективного использования.

ЛИТЕРАТУРА

1. **J.Dongarra et al.** DARPA HPCS Program: History, Models, Tools, Languages. – 2008
2. **Слуцкин А., Эйсымонт Л.** Российский суперкомпьютер с глобально адресуемой памятью. – Открытые системы, 2007, №9, с.42–51.
3. **Митрофанов В., Слуцкин А., Эйсымонт Л.** Суперкомпьютерные технологии для стратегически важных задач. – ЭЛЕКТРОНИКА: НТБ, 2008, №7, с.66–79.
4. **S. Scott et al.** Remote Translation Mechanism for a Multi-node System. US Patent 6922766, 2004.
5. **S. Scott et al.** The Cray BlackWidow: a Highly Scalable Vector Multiprocessor. – Proc. of the 2007 ACM/IEEE conference on Supercomputing, 2007.
6. **P. Konecny.** Introducing the Cray XMT. – Cray Inc., 2007.
7. **S. Scott, D. Abts, J. Kim, W. Dally.** The BlackWidow High-Radix Clos Network. – Stanford Univ., 2006.
8. **S.Kahan, P.Konecny.** A Memory Allocator for Multithreaded Architectures. – PPoPP, 2006.
9. **G.Alverson, C.Callahan, S.Kahan et al.** Synchronization techniques in a Multithreaded Environment. – US Patent 7117330, 2003.
10. **V.Tipparaju, A.Kot, J.Nierlocha et al.** Evaluation of Remote Memory Access Communication on the Cray XT3 /IPDPS 2007. IEEE International Volume Issue, 26–30 March 2007, pages 1–7.
11. **E.Strohmaier, H.Shan.** Apex-Map: A Global Data Access Benchmark to Analyze HPC Systems and Parallel Programming Paradigms. – Proceeding of the SC|05 Conference, 2005.
12. HPC Challenge Benchmarks. – http://icl.cs.utk.edu/hpcc
13. **Семенов А.С., Эйсымонт Л.К.** Параллельное умножение матриц на суперкомпьютере с мультитредово-потоковой архитектурой. – Программные системы и инструменты, №8. – М.: ВМиК МГУ.
14. **Семенов А.С.** Одномерное быстрое преобразование Фурье на суперкомпьютере с мультитредово-потоковой архитектурой/ Параллельные вычислительные технологии (ПаВТ'2008): Труды международной научной конференции (Санкт-Петербург, 28 января – 1 февраля 2008 г.). – ЮУрГУ, 2008, с. 224–231.
15. **Yoo A., Chow E., Henderson K., et al.** A Scalable Distributed Parallel Breadth-First Search Algorithm on BlueGene/L. – http://sc05.supercomputing.org/schedule/pdf/pap346.pdf
16. **Аверичева Д.Л., Семенов А.С., Фролов А.С.** Поиск вширь в графе на суперкомпьютере с мультитредово-потоковой архитектурой. – Информационные технологии, принята в печать.
17. **S.Alam, R.Barret, C.McCurdy et al.** Characterizing Applications on the Cray MTA-2 Multithreading Architecture. – Proc. of CUG Conf, 2006.