УДК 004.383.4

**Г. С. Речистов**, млад. науч. сотр., e-mail: grigory.rechistov@phystech.edu,
**А. А. Иванов**, науч. сотр.,
**П. Л. Шишпор**, млад. науч. сотр.,
**В. М. Пентковский**, д-р техн. наук, руководитель лаборатории,
Лаборатория суперкомпьютерных технологий для биомедицины,
фармакологии и малоразмерных структур,
Московский физико-технический институт

# Моделирование компьютерного кластера на распределенном симуляторе. Валидация моделей вычислительных узлов и сети

*Демонстрируется подход к задаче моделирования многопроцессорного кластера, состоящего из нескольких многоядерных компьютеров, соединенных высокопроизводительной сетью. Описываются используемый симулятор, окружение для проведения тестов и методы проведения измерений. Представлены результаты валидации моделей отдельных вычислительных узлов и сети, их соединяющей; предложены решения для преодоления обнаруженных ограничений и пути увеличения точности симуляции.*

***Ключевые слова:*** *распределенная симуляция, Simics, многоядерные системы, производительность симуляции, масштабируемость, кластер, Linpack*

## Введение

Расчеты, необходимые для современного научного исследования, могут занять неприемлемо большое время если проводить их на единственной ЭВМ, пусть даже максимально мощной и многопроцессорной. Другая опасность состоит в том, что не хватит емкости оперативной памяти или иных ресурсов для того, чтобы вместить все необходимые для работы данные. В таких случаях используется группа компьютеров, соединенных сетью, при этом задача разбивается и распределяется по этим узлам. Такая организация вычислений получила общепринятое название "кластер".

Перед создателями нового кластера зачастую стоит задача определения его оптимальной конфигурации, позволяющей достигнуть наибольшей производительности. При этом необходимо учитывать, какие конкретно приложения будут запускаться на этом кластере — они напрямую определяют требования на оборудование.

Для предсказания значений производительности создаваемой компьютерной системы удобной методикой является симуляция — моделирование еще не существующей аппаратуры с помощью программы, выполняющейся на уже доступных компьютерах. Данный подход используется на всех этапах разработки — от первых спецификаций и описания набора инструкций отдельного процессора до полной системы с работающей операционной системой и прикладными приложениями. Это позволяет обнаруживать ошибки проектирования на ранних этапах, снижая цену их исправления. Точность моделирования может варьироваться от высокой, демонстрируемой на сверхточных потактовых симуляторах, обязанных показывать поведение, неотличимое на уровне отдельных тактов от поведе-

ния настоящей системы, до достаточной в функциональных моделях, тем не менее способных загружать операционные системы и пользовательские приложения и при этом работать достаточно быстро. При симуляции многопроцессорных систем мы наталкиваемся на ряд следующих серьезных препятствий, обусловленных масштабами задачи:

- при последовательной симуляции всех моделируемых процессоров на одном реальном скорость работы будет ничтожно мала, поэтому необходимо использовать параллельные системы, что, в свою очередь, требует сложных схем синхронизации;
- ресурсов одной машины также может не хватить для содержания в себе модели целого кластера, поэтому модель сама по себе должна быть распределенной, т. е. выполняться на кластере.

Изучению и увеличению производительности приложений, использующих различные парадигмы многопроцессорных вычислений, посвящено множество работ. Описанию парадигмы передачи сообщений, такой как MPI, посвящены работы [1] и [2]; описанию систем с общей памятью и использующих OpenMP — работа [3]. Существуют также попытки создать адаптивные или гибридные системы, использующие лучшее из обоих подходов [4]. Моделированию суперкомпьютеров еще до их построения "в железе" посвящено несколько работ. Из них можно отметить BigSim [5] — симулятор IBM Blue Gene, а также MPI-SIM [6].

Данная работа описывает характеристики кластера, устанавливаемого для задач вычислительной биологии, симуляционную модель, построенную для анализа узких мест в его производительности и результаты измерений, выполненные на этой модели.

## Подход к моделированию кластера

Работа, описанная в данной статье, является лишь одним из этапов многоступенчатого плана: на существующем оборудовании создается модель будущего кластера, характеристики которого превосходят характеристики текущего в десятки раз. Когда будущий кластер воплощается в реальность, уже на нем строится модель следующего поколения. Это позволяет своевременно учитывать тенденции в развитии аппаратного обеспечения и уменьшить нагрузку на разработчиков модели.

## Характеристики моделируемого кластера

На рис. 1 приведена схема кластера, модель которого описана в данной работе. В табл. 1 приведены некоторые характеристики составляющих его компонентов.

Вычисления организованы следующим образом. Программа пользователя запускается на головном узле, затем она распределяется по вычислительным узлам, где и проводится большая часть вычислительной работы. Головной узел предоставляет сервисы координации общего хода работы, а также дисковое хранилище, доступное по протоколу NFS. Вычислительные узлы не имеют своего хранилища.

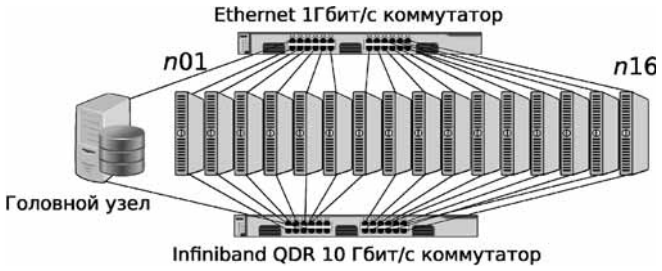


**Рис. 1. Схема кластера:**
*n*01—*n*16 — вычислительные узлы кластера

*Таблица 1*

**Параметры кластера**

| Параметр | Значение |
|---|---|
| Число вычислительных узлов | 16 |
| Процессоры узлов | Два Intel Xeon 5580, 3,33 ГГц, суммарно 12 ядер |
| Объем памяти головного узла | 48 Гбайт |
| Дисковое хранилище головного узла | 3 Тбайт |
| ОЗУ каждого вычислительного узла | 32 Гбайт DDR3 |
| Полное число вычислительных ядер | 192 |
| Полный объем ОЗУ для вычислений | 512 Гбайт |

## Используемый симулятор

Для построения модели был использован симулятор Simics [7]. Выбор был обусловлен его следующими достоинствами: полносистемная симуляция, учитывающая все аспекты работы как отдельного компьютера, так и системы связанных между собой узлов; высокая скорость функциональных моделей систем; возможность балансировать между скоростью работы и точностью во время симуляции; возможность сбора трасс событий для последующего детального изучения фактов, влияющих на производительность; множество доступных моделей устройств в поставке, позволяющих быстро создавать прототипы систем; легкость разработки и подключения новых моделей устройств — от центральных процессоров до сетевых карт, систем кэшей и криптографических ускорителей; наличие средств автоматизации процесса симуляции измерений с помощью сценариев.

Скорость и масштабируемость симуляции в Simics обеспечивается следующими возможностями:

- использованием двоичной трансляции [8] и аппаратных расширений виртуализации Intel VTx для достижения максимальной скорости работы (загрузка операционной системы внутри Simics может замедляться менее чем в 10 раз по сравнению с обычным ее исполнением);
- многопоточной симуляцией процессоров, позволяющей полностью использовать ресурсы ЭВМ;

- запуском программы в распределенном на несколько компьютеров режиме;
- технологией page sharing [9], используемой для экономии памяти.

## Замечания о точности используемых моделей

К сожалению, требования точности соответствия процесса и результатов симуляции реальному процессу и скорости работы самой модели часто противоречат друг другу. Авторами начата разработка полуаналитического метода для корректировки результатов, полученных в настоящей модели. При использовании трасс исполнения и характеристик аппаратуры, измеренных при помощи инструмента Intel VTune [10] или потактовых симуляторов, авторы планируют получать достаточно точные оценки производительности моделируемых систем.

В настоящей работе концентрируется внимание на двух аспектах производительности полной модели — процессоров и сетевых соединений.

Прежде всего, рассмотрим следующие особенности устройства ядра процессора, влияющие на производительность, без учета прилежащих к нему систем кэшей и памяти:

- современные процессоры Intel имеют так называемую out-of-order архитектуру, т. е. машинные инструкции могут исполняться в порядке, отличном от того, какой присутствует в памяти программы.
- наличие конвейера и нескольких декодирующих и вычислительных устройств позволяет эффективно выполнять несколько команд за такт.
- длинные и сложные команды архитектуры IA-32 разбиваются на более мелкие микрокоманды, которые могут быть исполнены в порядке, зависящем от текущего состояния вычислительного тракта процессора.

Существуют решения, моделирующие указанные выше аспекты микроархитектуры [11]. Однако необходимость учитывать такие подробности существенно снижает скорость самого симулятора.

Модели процессоров в Simics не учитывают out-of-order характер исполнения — инструкции обрабатываются в том порядке, в котором они найдены в памяти. Кроме того, по умолчанию на исполнение одной инструкции тратится ровно один такт симулируемого времени, тогда как в реальности длительность исполнения инструкции зависит от множества факторов, в первую очередь от типа этой команды. Это называется "функциональным моделированием".

В симуляторе Simics есть конфигурационный параметр, определяющий сколько тактов симулируемого процессора занимает исполнение одной инструкции. По умолчанию этот параметр равен 1. Для увеличения точности симулятора необходимо добавлять к системе отдельный модуль, определяющий задержки в циклах для каждой инструкции. Однако даже при использовании этого модуля вопрос об учете внеочередного исполнения команд остается открытым.

Последствия этого обстоятельства на результаты симуляции будут показаны далее, в секции результатов.

Simics предоставляет модели сетевых карт Ethernet для использования их в составе моделей полных вычислительных узлов. Устройства 1/10 Gigabit Ethernet поддерживаются используемой на модели кластера операционной системой GNU/Linux Debian 6, а также задействованы на реальной системе.

В Simics также есть простая модель сетевого коммутатора (*switch*), которая характеризуется значением задержки при передаче пакета и пропускной способностью. В модели также поддерживается режим с "бесконечной" пропускной способностью, в которой пакеты теряются лишь если они пришли в сеть одновременно с точностью до такта микропроцессора. Именно этот режим и использовался в экспериментах с целью определить оптимальную задержку и пропускную способность сетевого коммутатора.

## Тестовые задачи

Задачей данной работы являлось построение модели кластера и апробация методик измерения его производительности. Для этого был проведен ряд измерений на настоящей аппаратуре и на модели. Были использованы две следующие программы.

- *High Performance Linpack* [12] (далее Linpack). Данный тест, решающий систему линейных уравнений методом Гаусса, широко используется для оценки производительности систем на вычислениях с плавающей точкой. Кроме того, результаты, показанные на этом тесте, используются для составления списка Top 500 [13] суперкомпьютеров. В работе [14] подробно рассмотрены существующие варианты Linpack для архитектуры IA-32. В работе [15] описан алгоритм, используемый в данном тесте для решения системы уравнений.
- *Netperfmeter* [16]. Приложение для измерения скорости соединения по протоколам TCP, UDP и SCTP для Unix-систем. Позволяет задать сценарий, согласно которому в некоторые моменты времени будут создаваться и закрываться соединения. В конце работы приложение выводит статистику для прошедшего трафика. В этой работе программа использовалась для оценки максимальной скорости передачи данных между двумя компьютерами.

## Результаты измерений производительности узла на тесте Linpack

Данный раздел описывает результаты, полученные при верификации модели узла кластера, проведенной посредством сравнения результатов теста Linpack, полученных на реальной машине и на модели узла кластера. Отметим, что для достижения максимальной производительности на тесте Linpack необходима перекомпиляция этого теста для той машины, на которой будет проводиться запуск. Дело в том, что при компиляции тест оптимизируется в соответствии с ха-

рактеристиками используемой машины. Описанная процедура по сборке была проведена при запуске теста Linpack как на реальной машине, так и на симулируемой. Конфигурационный файл теста был использован один и тот же при всех запусках.

Linpack запускался в двух конфигурациях — на одном и на четырех ядрах. Значения размеров задачи $N$ брались в диапазоне от 100 до 32 000. При бóльших размерах задачи Linpack сообщал о невозможности выделить блок памяти нужного размера.

**Реальная ЭВМ.** Измерения проводились на системе со следующими характеристиками: процессор — Intel (R) Xeon (R) 5150 2,66 ГГц, 4 ядра, 16 Гбайт ОЗУ, операционная система Red Hat Enterprise Linux Server release 5.4 (x86_64).

На рис. 2 приведены графики зависимости производительности Linpack от размера задачи для одного и четырех ядер. В табл. 2 приведены значения четырех параметров для запусков: $P_{\max}$ — наибольшая продемонстрированная производительность, измеренная в гигафлопсах (флопс от англ. "*floating point operations per second*"); $N_{\max}$ — размер задачи, при которой достигнута $P_{\max}$; $N_{1/2}$ — размер задачи, при котором достигается половина от $P_{\max}$; $P_{peak}$ — пиковая, теоретическая производительность для машины, равная сумме числа инструкций над числами двойной точности на всех устройствах всех ядер системы, выполняемых за секунду работы. Также приведено значение эффективности системы, равное отношению максимальной производительности к теоретической: $P_{\max}/P_{peak}$.

**Моделируемая система.** Аналогичная зависимость была получена на модели вычислительного узла кластера. Измерения проводились на системе со следующими характеристиками: процессор Intel (R) Core i7 2,66 ГГц, 4 ядра, ОЗУ 16 Гбайт, ОС Debian 6.0.2 (64-битная).

На рис. 3 приведены графики зависимости производительности Linpack внутри симулятора от размера задачи для одного и четырех моделируемых ядер. В табл. 3 приведены значения $P_{\max}$, $N_{\max}$, $N_{1/2}$, $P_{peak}$ и эффективности для модели.

**Анализ результатов.** Полученные значения пиковой производительности для симулятора в четыре раза меньше аналогичных величин для реальной ЭВМ. Как уже было сказано, это вызвано различным значением числа команд, исполняемых за такт.

Другим интересным фактом является то, что достигнутая эффективность на симулируемой системе выше, чем на реальной: более 90 % против 80. Это объясняется отсутствием задержек, вызванных промахами системы кэшей — все доступы к памяти в модели всегда завершаются за один такт. В реальности это соответствовало бы системе, имеющей все свои данные

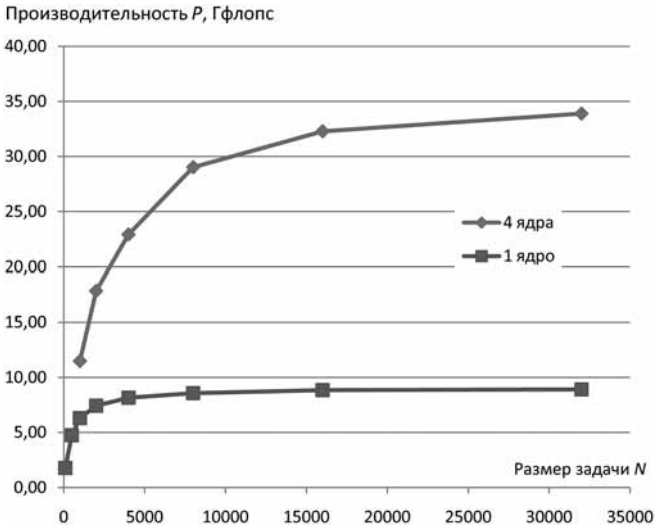


**Рис. 2. Измерения Linpack на реальной машине**

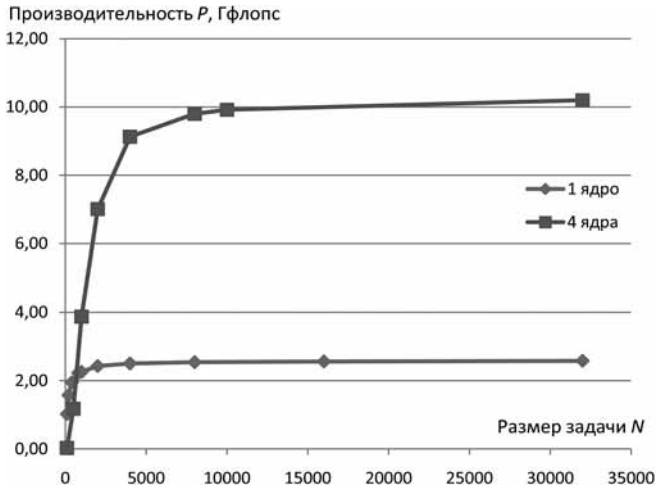


**Рис. 3. Измерения Linpack внутри симулятора**

*Таблица 2*

**Результаты прогона Linpack на реальной машине**

| Величина | Одно ядро | Четыре ядра |
|---|---|---|
| $P_{\max}$, Гфлопс | 8,91 | 33,9 |
| $N_{\max}$ | 32 000 | 32 000 |
| $N_{1/2}$ | 400 | 1800 |
| $P_{peak}$, Гфлопс | 10,64 | 42,56 |
| Эффективность, % | 83,74 | 79,65 |

*Таблица 3*

**Результаты прогона Linpack на симулируемой машине**

| Величина | Одно ядро | Четыре ядра |
|---|---|---|
| $P_{\max}$, Гфлопс | 2,58 | 10,1 |
| $N_{\max}$ | 32 000 | 32 000 |
| $N_{1/2}$ | 150 | 1200 |
| $P_{peak}$, Гфлопс | 2,66 | 10,64 |
| Эффективность, % | 97,00 | 94,92 |

(для $N = 32\,000$ это около 7,6 Гбайт) в кэше первого уровня. Таким образом, модель дает представление о верхней границе производительности для реальной системы с очень эффективным кэшом или очень быстрой памятью.

Построенная модель позволяет сравнивать между собой эффективность различных конфигураций, однако непосредственное сравнение реальных систем с симулируемыми требует пересчета результатов, учитывающего различное значение отношения "инструкций за такт".

## Результаты измерений на тесте Netperfmeter

Для тестирования использовались две машины (реальные в одном случае и симулируемые в другом). Командная строка запуска на активной стороне канала:

```
netperfmeter remotehost: 9000 –runtime 60
–tcp const0:exp2000:const0:exp2000:.
```

Это соответствует TCP-соединению со следующими характеристиками: полная длительность приема и передачи — 60 с, исходящее соединение — неограниченная (программно) скорость, исходящий поток имеет обратно-экспоненциальное распределение размеров пакетов со средним значением 2000 байт и максимальным 16 000 байт; входящий поток имеет аналогичные характеристики.

**Реальная система.** В физической системе были установлены сетевые карты Intel PRO/1000 PCI Gigabit Ethernet. Они были соединены через коммутатор с пропускной способностью 1 Гбит/с. Были получены следующие результаты:

- исходящий канал: скорость передачи данных 137 Мбайт/с;
- входящий канал: скорость передачи данных 110 Мбайт/с;
- потери: 267 кбайт/с.

**Моделируемая система.** В симулируемой системе использовалась модель сетевой карты Intel PRO/1000 PCI-Express Gigabit Ethernet. Полученные результаты:

- исходящий канал: скорость передачи данных 18 Мбайт/с;
- входящий канал: скорость передачи данных 8 Мбайт/с;
- потери: 19 кбайт/с.

**Анализ результатов.** Результаты тестирования соединения между реальными машинами показывают значения, близкие к теоретическому пределу (1000 Мбит/с = = 125 Мбайт/с). Это означает, что Netperfmeter может быть использован для адекватной оценки соединений.

В момент проведения тестов значение задержки сетевых пакетов было равно 400 мкс. Таким образом, для пакетов длиной 2000 байт эффективная пропускная способность равна приблизительно один пакет на один акт обмена, т. е. $2 \cdot 10^3$ байт/$4 \cdot 10^{-4}$ с = $5 \cdot 10^6$ байт/с = = 4,8 Мбайт/с, что по порядку величины совпадает с полученными ранее результатами.

Эти данные показали, что сетевая подсистема требовала дополнительной наладки. При исследовании причин низкой производительности оказалось, что максимально разрешенный размер пакета был равен лишь 2000 байтам, что существенно меньше возможностей гигабитной сетевой карты. Это обстоятельство было исправлено и последующие измерения работы Linpack проводились уже с большими пакетами.

## Результаты измерения производительности сети на тесте Linpack

В данном разделе описано измерение нагрузки на сетевой коммутатор в моделируемом кластере, а также влияние времени задержки пакетов на общее время исполнения Linpack. Измерения проводились на последовательных запусках Linpack с размером матрицы $N = 4000$ и параметрами распараллеливания задачи $P \times Q$ [12] $1 \times 192$, $6 \times 132$ и $16 \times 12$. Значение $N$ было выбрано максимально большим в условиях приемлемого времени исполнения тестов. $P \times Q$, равный $1 \times 192$, соответствует сетевому потоку, обеспечивающему минимальную нагрузку на коммутатор. Числа $6 \times 32$ и $16 \times 12$ были выбраны в предположении возрастании нагрузки на сеть. Точного описания параметров $P$ и $Q$ в руководстве по Linpack найдено не было, но был обнаружен совет, предлагающий варьировать $P \times Q$, чтобы достичь максимальной производительности при фиксированных параметрах сети.

Для надежности измерений Linpack запускался в режиме множественных экспериментов, где один и тот же тест исполнялся 4 раза, и только результаты последних трех испытаний принимались во внимание.

Параметры сетевого потока измерялись при помощи утилиты Wireshark [17]. Во всех результатах приводится средняя нагрузка, поскольку сетевой поток захватывался сразу после начала эксперимента и останавливался точно к концу теста. График зависимости потока от времени показал, что для всех экспериментов пиковая нагрузка незначительно превышала среднюю. Результаты измерений приведены в табл. 4.

**Влияние задержки на производительность Linpack.** В Simics значение задержки сетевого пакета связано с механизмом синхронизации времени в моделируемых узлах. Опуская излишние детали, отметим, что при уменьшении задержки частота синхронизации симулируемого времени между узлами модели кластера

*Таблица 4*

**Характеристики сетевого коммутатора при запуске теста Linpack на модели кластера**

| Параметры теста Linpack (при $N = 4000$) $P \times Q$ | Нагрузка, Мбит/с | Средний размер пакета, байты | Производительность, Гфлопс |
|---|---|---|---|
| $1 \times 192$ | 1300 | 5800 | 0,56 |
| $6 \times 32$ | 400 | 3700 | 1,1 |
| $16 \times 12$ | 150 | 1100 | 0,46 |

*Таблица 5*

**Влияние задержки сетевого пакета на производительность теста Linpack в модели кластера**

| Параметры теста Linpack (при $N = 4000$) $P \times Q$ | Производительность, Гфлопс, при задержке | | | | |
|---|---|---|---|---|---|
| | 80 мкс | 160 мкс | 240 мкс | 320 мкс | 400 мкс |
| 1 × 192 | 0,56 | 0,55 | 0,56 | 0,57 | 0,56 |
| 6 × 32 | 1,1 | 1,1 | 1,1 | 1,1 | 1,1 |
| 16 × 12 | 4,6 | 4,6 | 4,6 | 4,5 | 4,5 |

возрастает. Это может привести к значительному замедлению симуляции. В связи с этим было важно проверить, насколько влияет малое время задержки сетевого пакета на сообщаемую Linpack производительность. В табл. 5 показаны результаты измерений для различных выбранных значений задержки.

**Анализ результатов.** Модель показала, что нагрузка на сеть при работе Linpack даже при наибольших наблюдавшихся размерах пакетов много меньше (на четыре порядка) возможностей QDR Infiniband-коммутатора (т. е. для Linpack топология сети приемлема). Заслуживает внимания факт, что наблюдается различный размер транзакций между узлами сети при разных значениях $P \times Q$. Необходимо изучить, будет ли возрастать размер пакетов и, следовательно, нагрузка на сеть с ростом размера матрицы при бóльших значениях параметра $P$ теста Linpack.

Производительность Linpack не зависит от величины задержки сетевого пакета при значениях менее 400 мкс. Время обсчета очередного блока данных в Linpack больше времени сетевой транзакции, и это маскировало латентность сети. Этот факт позволил проводить все запуски Linpack для задержки сетевого пакета в 400 мкс, что значительно ускорило процесс симуляции.

## Заключение

Результаты исследований, проведенных на отдельных узлах, а также на полной модели кластера, показали соответствие реальных и моделируемых характеристик ЭВМ и пригодность выбранного подхода для изучения характеристик его производительности на простой функциональной модели. При этом необходимо делать поправку на обнаруженные ограничения и разрабатывать метод коррекции результатов для получения достаточно точных значений производительности. Разрабатывается методология оценки производительности с использованием трасс симуляции и параметрами ЦПУ, памяти, полученными при помощи VTune или потактовых моделей, подключаются модели кэшей, уточняется набор трассируемых событий.

Большой практический интерес представляет изучение характера работы разных пользовательских приложений на модели кластера. В нашем случае это приложения молекулярной динамики. Выработанные на основе экспериментов с Linpack и Netperfmeter методики будут адаптированы к более сложным приложениям, основанным на пакете Gromacs [18].

Следует отметить, что нашей целью является не только исследование кластера в его текущей конфигурации, но и поиск оптимальных конфигураций с учетом его развития (в 2012 г. пиковая производительность кластера будет наращена в несколько раз путем увеличения числа входящих в него вычислительных узлов). Использование симулятора для оценок производительности выглядит многообещающе — результаты симуляции должны позволить вырабатывать рекомендации по оптимальному развитию кластера.

### Список литературы


1. **Demaine E.** A threads-only MPI implementation for the development of parallel programs // Proc. of HPCS97. Winnipeg, Manitoba, Canada. 1997. P. 153—163.

2. **Riesen R.** A Hybrid MPI Simulator // Proc. of IEEE International Conference on Cluster Computing. Barcelona, Spain. 2006. P. 1—9.

3. **Hu Y. C., Honghui L., Cox A., Willy Z.** OpenMP for networks of SMPs // Journal of Parallel and Distributed Computing. 2000. Vol. 60. P. 1512—1530.

4. **Carribault P., Perache M., Jourdren H.** Enabling low-overhead hybrid MPI/OpenMP parallelism with MPC // Proc. of the 6th International Workshop on OpenMP. Tsukuba, Japan. 2010. P. 80—93.

5. **Zheng G., Kakulapati G., Kale L. V.** BigSim: A parallel simulator for performance prediction of extremely large parallel machines // Proc. of Parallel and Distributed Processing Symposium. Santa Fe, New Mexico, USA. 2004. Vol. 1. P. 78.

6. **Prakash S., Bagrodia R. L.** MPI-SIM: using parallel simulation to evaluate MPI programs // Proc. of the Winter Simulation Conference. Washington DC. 1998. P. 467—474.

7. **Magnusson P.S., Christensson M., Eskilson J.** et al. Simics: A full system simulation platform // Computer. 2002. Vol. 35. P. 50—58.

8. **Sites R. L., Chernoff A., Kirket M. B.** et al. Binary translation // Communications of the ACM. 1993. Vol. 36. N 2. P. 69—81.

9. **Bugnion E., Devine S., Rosenblum M**. Disco: Running commodity operating systems on scalable multiprocessors // Proc. of the sixteenth ACM Symposium on operating systems principles. 1997. P. 143—156.

10. **Intel** Corporation. Intel® VTune Performance Analyzer 9.1 for Linux* — Documentation. URL: http://software.intel.com/en-us/articles/intel-vtune-performance-analyzer-for-linux-documentation.

11. **Matt Y. T.** PTLsim User's Guide and Reference. 2007. URL: http://www.ptlsim.org/Documentation/PTLsimManual.pdf.

12. **High** performance Linpack benchmark. URL: http://www.netlib.org/Linpack/.

13. **TOP500** supercomputing sites. URL: http://www.top500.org.

14. **Речистов Г. С.** Бенчмарк Linpack для архитектуры Intel IA-32: существующие варианты, сравнительный анализ возможностей и демонстрируемой производительности // Труды 54-й научной конференции МФТИ. Изд-во МФТИ, 2011. Т. 1. С. 71—72.

15. **Dongarra J. J.**, **Luszczek P., Petitet A.** The Linpack benchmark: Past, present, and future // Concurrency and Computation: Practice and Experience. 2003. Vol. 15. P. 20.

16. **Dreibholz T.** Netperfmeter: A TCP/UDP/SCTP/DCCP network performance meter tool. URL: http://www.iem.uni-due.de/~dreibh/netperfmeter/.

17. **Orebaugh A.**, **Ramirez G., Burke J.** Wireshark & Ethereal Network Protocol Analyzer Toolkit. Syngress Publishing, 2007. 540 p.

18. **Van Der Spoel D.,** Lindahl E. et al. Hess B. GROMACS: Fast, exible, and free // Journal of Computational Chemistry. 2005. Vol. 26. N. 16. P. 1701—1718.