Моделирование компьютерного кластера на распределённом симуляторе

Верификация моделей вычислительных узлов и сети кластера

Речистов Григорий, Александр Иванов, Павел Шишпор
Факультет радиотехники и кибернетики
МФТИ
Москва, Россия
grigory.rechistov@phystech.edu, alexander.a.ivanov@phystech.edu, pavel.shishpor@phystech.edu

Abstract: **In this paper we present an approach to modeling of large many cores cluster system consisting of several multi-core computers connected with high speed networks. A simulation solution, testing environment and methods of running tests are described. We give results of verification of computing nodes and interconnect model against their real counterparts and propose several solutions to overcome limitations discovered and ways of increasing the accuracy.**

Key words: distributed simulation, Simics, multi core systems, simulation performance, scalability, cluster, linpack.

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

Ключевые слова: распределённая симуляция, Simics, многоядерные системы, производительность симуляции, масштабируемость, кластер, linpack.

I. Введение

Расчёты, необходимые для современного научного исследования, могут занять неприемлемо большое время, если производить их на единственной ЭВМ, пусть даже максимально мощной и многопроцессорной. Иная опасность состоит в том, что не хватит ёмкости оперативной памяти или иных ресурсов для того, чтобы вместить все необходимые для работы данные. В таких случаях используют группу компьютеров, соединённых сетью, при этом задачу разбивают и распределяют по этим узлам. Такая организация вычислений получила общепринятое название "кластер".

Перед создателями нового кластера зачастую стоит задача определения его оптимальной конфигурации, позволяющей достигнуть значений производительности, близких к теоретически возможной сумме производительности узлов, входящих в его состав. 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 ещё не существующей аппаратуры с помощью программы, выполняющейся на уже доступных компьютерах. Данных 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 практически полной, демонстрируемой на сверхточных потактовых симуляторах, обязанных показывать поведение, неотличимое на уровне отдельных тактов от поведения настоящей системы, до достаточной в функциональных симуляторах, тем не менее способных загружать операционные системы и пользовательские приложения и при этом работающие достаточно быстро для процесса прототипирования.

При симуляции многопроцессорных систем мы наталкиваемся на ряд серьёзных 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 распределённой, т.е. выполняться на кластере.

Кроме того, всегда необходимо понимать, насколько точно симулятор соответствует той системе, которую он призван описывать, и насколько точны измерения, полученные с его помощью.

Работа выполнена в рамках гранта, выделенного в соответствии с постановлением Правительства России №220 от 9.04.2010 г.

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

Данная статья организована следующим образом. В секции II приведены ссылки на работы, посвящённые исследованиям производительности многопроцессорных систем, а также симуляции кластеров и систем с большим числом вычислительных процессоров. Секция III рассказывает о долгосрочной программе по созданию высоко производительного кластера. Секция IV содержит описание кластера, характеристики которого использовались для создания модели. Секция V рассказывает о симуляторе Simics и причинах выбора его для создания модели. Секция VI описывает отличия моделей от реальных устройств. Секция VII описывает использованные для верификации тестовые приложения. Секции VIII - X описывают результаты проведённых измерений. Секция XI заключаёт работу и очерчивает направления наших последующих исследований.

II. Обзор литературы

Изучению и увеличению производительности приложений, использующих различные парадигмы многопроцессорных вычислений, посвящено множество работ. Для парадигмы передачи сообщений, такой как MPI, можно отметить работы [1] и [2]; для систем с общей памятью и использующих OpenMP [3]. Существуют также попытки создать адаптивные или гибридные системы, использующие лучшее из обоих подходов [4].

Моделированию суперкомпьютеров ещё до их построения «в железе» посвящено несколько работ. Из них можно отметить BigSim [5] — симулятор IBM Blue Gene, а так же MPI-SIM [6].

III. Подход к моделированию кластера

Несмотря на широкие возможности моделирования, сразу же построить «идеальный» виртуальный кластер нельзя. Так получается по той простой причине, что сама модель должна исполняться на реальном оборудовании, потенциал которого не безграничен. Поэтому работа, описанная в данной статье является лишь одним из этапов многоступенчатого плана.

Суть его состоит в том, что бы на существующем оборудовании строить модель будущего кластера, характеристики которого превосходят текущий кластер в десятки раз. Когда будущий кластер воплощается в реальность уже на нем строится модель следующего поколения. Такой подход позволяет своевременно учитывать тенденции в развитии аппаратного обеспечения и уменьшить нагрузку на разработчиков модели.

IV. Характеристики моделируемого кластера

На рис. 1 приведена схема кластера модель которога описана в данной работе. В табл. 1 приведены некоторые численные 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.

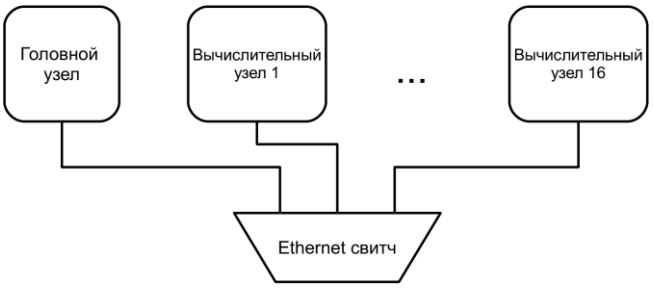

Рис. 1. Схема кластера.

ТАБЛИЦА I. Параметры кластера.

Параметр	Значение
Число вычислительных узлов	16
Процессор головного узла	Два шестиядерных Intel Xeon 5580, 3,33 ГГц
Объём памяти головного узла	48 Гбайт
Дисковое хранилище головного узла	3 Тбайт
Процессоры вычислительных узлов	Два шестиядерных Intel Xeon X5580, 3,33 ГГц
Объём памяти каждого вычисл. Узла	32 Гбайт
Полное число вычислительных ядер	192
Полный объём памяти для вычислений	512 Гбайт

V. Используемый симулятор

Для построения модели был использован симулятор Simics [7]. Наш выбор был обусловлен следующими особенностями программы, делающей её подходящей для решения нашей задачи:

- Полносистемная симуляция, позволяющая учесть 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; например, модели кэшей, в общем случае сильно

замедляющих симуляцию, можно динамически включать и выключать, когда они не нужны.

- Возможность сбора аппаратных трасс, что позволяет использовать результаты прогона модели передавать для более точного анализа производительности в потактовую модель.
- Множество доступных моделей устройств в поставке, позволяющих быстро создавать прототипы систем.
- Лёгкость разработки и подключения новых моделей устройств — от центральных процессоров до сетевых карт, систем кэшей и криптографических ускорителей.
- Средства автоматизации процесса симуляции измерений с помощью сценариев.

Скорость и масштабируемость симуляции в Simics обеспечивается следующими возможностями:

- Использование двоичной трансляции [8] и аппаратных расширений виртуализации Intel VTx для достижения максимальной скорости работы (так, загрузка операционной системы внутри Simics может замедляться менее чем в 10 раз по сравнению с обычным её исполнением.)
- Многопоточная симуляция процессоров, позволяющая полностью использовать ресурсы ЭВМ.
- Возможность запускать программу в распределённом на несколько компьютеров режиме.
- Для экономии памяти идентичные страницы моделируемой памяти (даже из независимых машин) сливаются в одну (технология page sharing [9]).

VI. Замечания о точности используемых моделей

К сожалению, требования точности соответствия процесса и результатов симуляции реальному процессу и скорости работы самой программы-симулятора практически всегда противоречат друг другу.

Сейчас в рамках проекта начата разработка полу-аналитического метода для корректировки результов, полученных в настоящей модели. При использовании трасс исполнения и характеристик аппаратуры, измеренных при помощи VTune или потактовых симуляторов, авторы планируют получать достаточно точные оценки производительности моделируемых систем.

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

A. Моделирование центрального процессора

Прежде всего, рассмотрим работу непосредственно ядра процессора, без учёта прилежащих к нему систем кэшей и памяти.

- Современные процессоры Intel имеют т. н. 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 [10]. Однако необходимость учитывать микроархитектурные 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 это 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 учёте внеочередного исполнения команд остаётся открытым.

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

B. Сеть

Simics предоставляет простую модель сетевых устройств Ethernet, довольно точно отражающие работу реальных карт. Среди моделей есть устройства Gigabit Ethernet и 10 Gigabit Ethernet, распознаваемые и поддерживаемые использованной нами операционной системой (GNU/Linux Debian 6), запущенной на модели кластера.

В Simics так же есть простая модель сетевого коммутатора (switch) которая параметризуется величиной задержки при передачи 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.

VII. Тестовые задачи

Задачей данной работы являлось построение модели кластера и опробация методик измерения его производительности. Для этого был произведён ряд измерений на настоящей аппаратуре, а затем аналогичные тесты были прогнаны на модели. Для измерений мы использовали следующие две программы.

- High Performance Linpack [11]. Данный тест, решающий систему линейных уравнений методом Гаусса, широко используется для оценки производительности вычислительных комплексов на вычислениях с плавающей точкой. Кроме того, результаты, показанные на этом тесте, используются для составления списка Top 500 [12] суперкомпьютеров. В работе [13] подробно рассматриваются существующие варианты LINPACK для архитектуры IA-32. В [14] описывается алгоритм и его параметры.
- Netperfmeter [15]. 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 выводит статистику для прошедшего траффика. В этой работе программа использовалась для оценки максимальной скорости передачи данных между двумя компьютерами.

VIII. Результаты измерений производительности узла на тесте Linpack

Данный раздел описывает результаты, полученные при верификации модели узла кластера, проведённой 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 производиться запуск. Дело в том, что при компиляции тест оптимизируется в соответствии с характеристиками используемой машины. Описанная процедура по сборке была проведена при запуске теста Linpack как на реальной машине, так и на симулируемой. Конфигурационный файл теста LINPACK был использован один и тот же при всех запусках.

Linpack запускался в двух конфигурациях – на 1 и на 4 ядрах. Значения размеров задачи N брались в диапазоне от 100 до 32000. При бо́льших размерах задачи Linpack сообщал о невозможности выделить блок памяти нужного размера.

A. *Реальная ЭВМ*

Измерения производились на системе со следующими характеристиками:

- Процессор — Intel (R) Xeon (R) 5150 2,66 ГГц, 4 ядра.
- Память — 16 Гбайт.
- Операционная система — Red Hat Enterprise Linux Server release 5.4 (x86_64).

На рис. 2 приведёны графики зависимости производительности Linpack от размера задачи для 1 и 4 ядер.

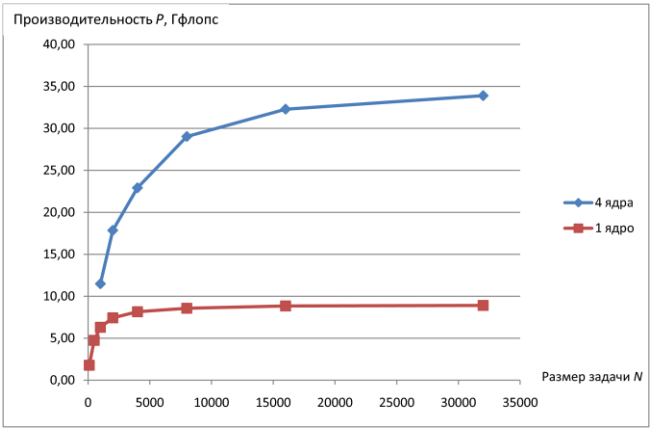

Рис. 2. Измерения LINPACK на реальной машине на одном и четырёх ядрах.

В таблице 2 приведены значения четырёх параметров для запусков. Здесь P_{max} – наибольшая продемонстрированная производительность, измеренная во гигафлопсах (от англ. "floating point operations per second"), N_{max} – размер задачи, при которой она достигнута, $N_{1/2}$ – 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}.

ТАБЛИЦА II. Результаты прогона Linpack на реальной машине.

Величина	1 ядро	4 ядра
P_{max}, Гфлопс	8,91	33,9
N_{max}	32000	32000
$N_{1/2}$	400	1800
P_{peak}, Гфлопс	10,64	42,56
Эффективность	83,74%	79,65%

B. *Моделируемая система*

Аналогичная зависимость была получена на модели вычислительного узла кластера.

Измерения производились на системе со следующими характеристиками:

- Процессор — Intel (R) Core i7 2,66 ГГц, 4 ядра.
- Память — 16 Гбайт.
- Операционная система — Debian 6.0.2 (x86_64).

На рис. 3 приведёны графики зависимости производительности Linpack внутри симулятора от размера задачи для 1 и 4 ядер.

В таблице 3 приведены значения P_{max}, N_{max}, $N_{1/2}$, P_{peak} и эффективности для симулируемой модели.

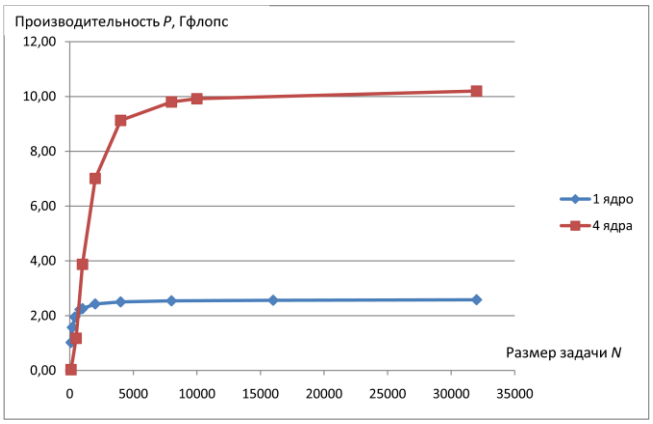

Рис. 3. Измерения LINPACK внутри симулятора.

ТАБЛИЦА III. РЕЗУЛЬТАТЫ ПРОГОНА LINPACK НА СИМУЛИРУЕМОЙ МАШИНЕ.

Величина	1 ядро	4 ядра
P_{max}, Гфлопс	2,58	10,1
N_{max}	32000	32000
$N_{1/2}$	150	1200
P_{peak}, Гфлопс	2,66	10,64
Эффективность	97,00%	94,92%

C. *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 процентов 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 (для N = 32000 это около 7,6 Гбайт) в кэше первого уровня. Таким образом, модель даёт представление о верхней границе производительности для реальной системы с очень эффективным кэшом или очень быстрой памятью.

В результате был сделан вывод, что построенная модель позволяет сравнивать между собой эффективность различных конфигураций, однако непосредственное сравнение реальных систем с симулируемыми требует пересчёта результатов, учитывающего различное значение отношения "инструкций за такт".

IX. 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:
description="Sample TCP"
```

Это соответствует TCP-соединению со следующими характеристиками

- Полная длительность приёма и передачи – 60 секунд.
- Исходящее соединение – неограниченная (программно) скорость.
- Характеристики исходящего траффика: обратно-экспоненциальное распределение, средний размер пакета 2000 байт, максимальный – 16000 байт.
- Входящий траффик имеет характеристики, совпадающие с параметрами исходящего.

A. *Реальная система*

В физической системе были установлены сетевые карты Intel PRO/1000 PCI Gigabit Ethernet. Они были соединены через свитч с пропускной способностью 1 Гбит/с. Были получены следущие числа.

- Исходящий канал: скорость передачи данных 137 Мбайт/с.
- Входящий канал: скорость передачи данных 110 Мбайт/с.
- Потери: 267 кбайт/с.

B. *Моделируемая система*

В симулируемой системе использовалась модель сетевой карты Intel PRO/1000 PCI-Express Gigabit Ethernet.

- Исходящий канал: скорость передачи данных 18 Мбайт/с.
- Входящий канал: скорость передачи данных 8 Мбайт/с.

- Потери: 19 кбайт/с.

C. *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 симулируемым микросекундам. 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 результатами бенчмарка.

Полученные числа показывают, что модель сетевого взаимодействия требовала наладки. В частности было выяснено, что максимально разрешенный размер пакета был всего лишь 2000 байт, что существенно меньше возможностей гигабитной сетевой карты.

Дальнейшие измерения на задаче Linpack проводились уже с большими пакетами.

X. Результаты измерения производительности сети на тесте Linpack

В данном разделе описано измерение нагрузки на сетевой коммутатор в моделируемом кластере, а так же влияние времени задержки пакетов на общее время исполнения Linpack.

Измерения производились на последовательных запусках Linpack с размером матрицы *N* = 4000 и параметрами распараллеливания задачи PxQ [11] 1x192, 6x132 и 16x12. Значение *N* было выбрано максимально большим в условиях приемлемого времени исполнения тестов. PxQ, равный 1x192, соответствует сетевому потоку, обеспечивающему минимальную нагрузку на коммутатор. Числа 6x32 и 16x12 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 PxQ, чтобы достичь максимальной производительности при фиксированых параметрах сети.

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

Параметры сетевого потока измерялись при помощи утилиты “Wireshark” [16].

A. *Нагрузка на сетевой коммутатор*

Во всех результатах приводится средняя нагрузка, поскольку сетевой поток захватывался сразу после начала эксперимента и останавливался точно к концу теста. График величины потока от времени показал, что для всех экспериментов пиковая нагрузка незначительно превышала среднюю.

ТАБЛИЦА IV. Нагрузка на сетевой коммутатор при запуске Linpack на модели кластера.

Linpack: N, PxQ	**Нагрузка (Mbit/s)**	**Средний размер пакета (Bytes)**	**Gflops**
4000, 1x192	1300	5800	0,56
4000, 6x32	400	3700	1,1
4000, 16x12	150	1100	0,46

B. *Влияние задержки на производительность Linpack*

В Simics величина задержки сетевого пакета связана с механизмом синхронизации времени в моделируемых узлах. Опуская излишние детали, отметим, что при уменьшении величины задержки частота синхронизации симулируемого времени между узлами модели кластера возрастает. Это может привести к значительному замедлению симуляции. В связи с этим было важно проверить, насколько влияет малое время задержки сетевого пакета на рапортуемую Linpack-ом производительность. ТАБЛИЦА V. показывает результаты измеренений для различных выбранных значений задержки.

ТАБЛИЦА V. Влияние задержки сетевого пакета на производительность Linpack в модели кластера.

Linpack N, PxQ	**Gflops, Delay= 80мкс**	**Gflops, Delay= 160мкс**	**Gflops, Delay= 240мкс**	**Gflops, Delay= 320мкс**	**Gflops, Delay= 400мкс**
4000, 1x192	0,56	0,55	0,56	0,57	0,56
4000, 6x32	1,1	1,1	1,1	1,1	1,1
4000, 16x12	4,6	4,6	4,6	4,5	4,5

C. *Анализ результатов*

Модель показала, что нагрузка на сеть в задаче Linpack даже при наибольших наблюдавшихся размерах пакетов много меньше (на 4 порядка) возможностей QDR Infiniband коммутатора (т.е. для задачи Linpack топология сети более чем приемлема).

Также заслуживает внимания факт, что наблюдается различный размер транзакций между узлами сети при разных PxQ. Необходимо изучить будет ли возрастать размер пакетов, и, следовательно, нагрузка на сеть, с ростом размера матрицы при б**о**льших значениях параметра “P” задачи Linpack.

Выяснилось, что производительность Linpack не зависит от величины задержки сетевого пакета менее 400 микросекунд. По-видимому время обсчёта очередного блока данных в Linpack больше времени сетевой транзакции, и это маскировало латентность сети. Этот факт позволил нам проводить все тесты на Linpack для величины задержки сетевого пакета в 400 мкс, что

значительно ускорило процесс симуляции (см. «Влияние задержки на производительность Linpack» выше).

XI. Заключение и направления дальнейшего исследования

Результаты исследований, проведённых на отдельных узлах, а также на полной модели кластера, показали соответсвие 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. Данная работа уже начата:

- 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 адаптированны к более сложным приложениям, основанным на математическом пакете Gromacs [17] или на других библиотеках, используемых в биологическом моделировании.

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 путём увеличения числа входящих в него вычислительных узлов). Использование симулятора для оценок производительности выглядит многообещающим — результаты симуляции должны позволить нам вырабатывать рекомендации по наиболее оптимальному развитию кластера.

Список литературы

[1] Demaine, Erik. A threads-only MPI implementation for the development of parallel programs / Erik. Demaine. — 1997.—July. — Pp. 153–163. http://erikdemaine.org/software/TOMPI/.

[2] Riesen, Rolf. A hybrid MPI simulator / Rolf. Riesen // CLUSTER.— 2006.

[3] Hu, Y. Charlie. OpenMP for networks of SMPs. — 1999.

[4] Enabling low-overhead hybrid MPI/OpenMP parallelism with MPC / Patrick. Carribault, Marc. Perache, Herve. Jourdren // 6th International workshop on OpenMP. — 2010.—June. http://mpc.sourceforge.net/files/CarPerJou_IWOMP2010_LNCS.pdf.

[5] Zheng, Gengbin. BigSim: A parallel simulator for performance prediction of extremely large parallel machines / Gengbin. Zheng, Gunavardhan. Kakulapati, Laxmikant V. Kale // Parallel and Distributed Processing Symposium, International. — 2004. — Vol. 1. — P. 78b. http://charm.cs.uiuc.edu/research/bigsim/.

[6] Prakash, Sundeep. MPI-SIM: using parallel simulation to evaluate MPI programs / Sundeep. Prakash, Rajive L. Bagrodia //Winter Simulation Conference. — 1998. — Pp. 467–474.

[7] Simics: A full system simulation platform / Peter S. Magnusson, Magnus. Christensson, Jesper. Eskilson et al. // Computer. — 2002.— February. — Vol. 35. — Pp. 50–58. http://portal.acm.org/citation.cfm?id=619072.621909.

[8] Binary translation / Richard L. Sites, Anton. Chernoff, Matthew B. Kirk et al. // Communications of the ACM. — 1993.—February. — Vol. 36, no. 2. — Pp. 69–81.

[9] Bugnion, Edouard. Disco: Running commodity operating systems on scalable multiprocessors / Edouard. Bugnion, Scott. Devine, Mendel. Rosenblum // ACM Transactions on Computer Systems.—1997. — Pp. 143–156. http://www.cis.upenn.edu/~cis700-6/04f/papers/bugnion-disco.pdf.

[10] Yourst, Matt T. PTLsim user's guide and reference. — 2007.

[11] High performance Linpack benchmark. http://www.netlib.org/linpack/.

[12] TOP500 supercomputing sites. http://www.top500.org.

[13] Бенчмарк LINPACK для архитектуры Intel IA-32: существующие варианты, сравнительный анализ возможностей и демонстрируемой производительности — 2011. — Неопубликовано. http://atakua.doesntexist.org/public/linpack-survey.pdf

[14] Dongarra, Jack J. The LINPACK benchmark: Past, present, and future. / Jack J. Dongarra, Piotr. Luszczek, Antoine. Petitet // Concurrency and Computation: Practice and Experience. — 2003. — Vol. 15. — P. 2003.

[15] Dreibholz, Thomas. Netperfmeter: A TCP/UDP/SCTP/DCCP network performance meter tool. http://www.iem.uni-due.de/~dreibh/netperfmeter/.

[16] Wireshark & Ethereal Network Protocol Analyzer Toolkit (Jay Beale's Open Source Security) / Angela. Orebaugh, Gilbert. Ramirez, Josh. Burke, Larry. Pesce. — Syngress Publishing, 2006.

[17] GROMACS: Fast, flexible, and free / David. Van Der Spoel, Erik. Lindahl, Berk. Hess et al. // Journal of Computational Chemistry.— 2005. — Vol. 26, no. 16. — Pp. 1701–1718. http://dx.doi.org/10.1002/jcc.20291.