УДК 004.054

А.А. Бородин

ИССЛЕДОВАНИЕ НАГРУЗОЧНЫХ СПОСОБНОСТЕЙ КОМПЬЮТЕРНЫХ СИСТЕМ

A.A. Borodin

RESEARCH OF COMPUTER SYSTEMS LOAD CAPABILITY

Нагрузочное тестирование является одним из основных способов измерения характеристик и проверки корректности функционирования информационных систем при эксплуатационных условиях. Ввиду роста и усложнения информационных систем для их тестирования требуется применение все большей нагрузки. Для ее создания оказывается недостаточно ресурсов одного компьютера, поэтому применяются механизмы, основанные на совокупности объединенных компьютерных узлов. Приведены результаты экспериментов по определению предельных нагрузочных способностей одиночных компьютерных систем, а также факторов, оказывающих на них влияние.

НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ; НАГРУЗОЧНЫЕ СПОСОБНОСТИ; ВРЕМЯ ОБРАБОТКИ ПАКЕТА ОПЕРАЦИОННОЙ СИСТЕМОЙ.

Load testing is one of the main tools used to measure parameters and determine computer system behavior under operating conditions. Constant development and extension of computer systems lead to the load increase to maintain normal condition. As the resources of a single computer are not enough to create that amount of load, there are different methods to reach that amount of load using distributed computing. This article presents the experiment results to determine load capability limits for a single computer system and factors affecting them.

LOAD TESTING; LOAD CAPABILITY; PROCESSING TIME OF A PACKET BY OPERATING SYSTEM.

Нагрузочное тестирование является важнейшим средством, с помощью которого проверяется корректность функционирования и измерение характеристик информационных систем в эксплуатационных условиях. Эффективность проведенного нагрузочного тестирования напрямую влияет на качество разрабатываемой информационной системы.

Процесс тестирования состоит из множества шагов [1], которые условно можно разделить на три стадии: подготовка к тестированию, запуск тестов (сбор результатов) и анализ результатов. Основные усилия большинства исследователей направлены на решение вопросов эффективной подготовки к тестированию [2–9] и анализа результатов [10, 11]. Однако вопросам запуска тестов уделяется значительно меньшее внимание. При тестировании используются готовые программные продукты (инструменты нагрузочного тестирования). В настоящее время появилось большое количество подобных программ – проприетарных [12] и свободных [13]. Все они при выполнении стадии запуска тестов (создания нагрузки на информационную систему) используют ресурсы обычных компьютерных узлов. В случае их нехватки для наращивания создаваемой нагрузки на тестируемую систе-

му, применяют облачные и кластерные вычисления или совокупность компьютеров, объединенных с помощью локальной сети. Однако применение облачных и кластерных вычислений требует подключения к глобальной сети, что не всегда рационально и возможно. Поэтому информационную систему в большинстве случаев тестируют при помощи компьютеров, объединенных в локальную сеть. Постоянное усложнение информационных систем требует применения все большего количества компьютерных узлов для достижения эксплуатационных условий. Это приводит к увеличению затрат и усилий на обслуживание такой системы. Именно на преодоление этих недостатков и направлено наше исследование.

В настоящее время стадия запуска тестов реализовывается в виде модуля нагрузки. Он представляет собой одну из подпрограмм инструментов нагрузочного тестирования. Сформированные на подготовительных этапах тесты передаются этому модулю. На основании тестов модуль нагрузки формирует большое количество вызовов функций интерфейса программирования приложений операционной системы. Итогом выполнения каждой такой функции является отправка в сеть сформированного модулем нагрузки запроса. После того как модуль выполнит вызов функции, задачу отправки запроса выполняет операционная система. Для выполнения этой задачи ОС проводит запуск подпрограмм, которые осуществляют подготовку запроса к отправке, помещают запрос в пакет, определяют путь его следования и проводят передачу сформированного пакета в контроллер среды передачи. Операционная система может запускать и ряд других подпрограмм, например, осуществляющих проверку допустимости отправки запроса в соответствии с настройками безопасности ОС. Исполнение этого множества подпрограмм ложится на центральный процессор. По мере увеличения количества вызовов функций интерфейса программирования, подпрограммами ОС затрачивается все большее количество ресурсов центрального процессора. Это не сильно влияло бы на количество сформированных запросов в секунду, если был запущен лишь модуль нагрузки. Однако в большинстве случаев в ОС запущено большое количество сторонних прикладных приложений, которые выполняют свои функции. Одновременно с этим при выполнении шагов создания нагрузки должны быть запущены программы, осуществляющие анализ поступающих ответов от тестируемой системы. Эти приложения собирают наиболее ценные данные нагрузочного тестирования – метрики производительности. Именно они подвергаются анализу, и на основе их делаются выводы о качестве спроектированной информационной системы. Для выполнения всей этой работы ресурсов одного компьютерного узла не хватает.

Предельная интенсивность генерации сообщений компьютерного узла зависит от архитектуры сетевого адаптера – устройства, обеспечивающего связь компьютера с сетью [14]. Возможность достижения этой интенсивности зависит от операционной системы, запущенных приложений и архитектуры самого компьютера. Мы провели ряд экспериментов с целью измерения времени, необходимого операционной системе для обработки отправляемого в сеть сообщения.

Для проведения экспериментов была использована программа tcpdump, входящая в состав UNIX-подобных операционных систем. Она использует низкоуровневую библиотеку pcap (packetcapture), осуществляющую перехват данных на уровне драйвера в соответствии с назначенными параметрами [15]. Каждый перехваченный пакет снабжается временной меткой. Она указывает системное время ОС, когда пакет был перехвачен. Для измерения времени, необходимого ОС для обработки сообщения, выполнялись следующие действия:

считывание системного времени;

помещение его в пакет;

отправка пакета через сетевой стек;

перехват пакета;

вычисление искомого времени, исходя из разницы между временем отправки и временем перехвата.

Специально созданная экспериментальная программа отправляла системное время

в сеть с помощью транспортного протокола UDP. Этот протокол был выбран потому, что операционной системе требуется меньшее количество вычислительных ресурсов на его обработку. В программе было предусмотрено постепенное увеличение количества передаваемых сообщений в секунду, что позволило отследить влияние интенсивности генерации сообщений на время обработки одного сообщения. Для сбора данных применялась программа tcpdump, встроенная в операционные системы всех экспериментальных платформ. Эксперимент состоял из следующих шагов:

- запуск программы перехвата пакетов tcpdump;
- запуск экспериментальной программы;
- ожидание завершения работы экспериментальной программы;
- завершение работы программы перехвата пакетов;
- сбор и анализ результатов.

На практике в ходе проведения нагрузочного тестирования применяются обычные компьютерные узлы под управлением различных операционных систем. По этой причине, для достоверности результатов, эксперименты проводились также на базе нескольких обычных компьютерных узлов с актуальным аппаратным и программным обеспечением. В нашем исследовании эти узлы называются *экспериментальными платформами* (ЭП). Во время эксперимента применялись следующие компьютерные узлы:

экспериментальная платформа 1, компьютер Intel Core i7 920 2.67 ГГц, DDR3-1066 9 Гб, CentOS 6.4;

экспериментальная платформа 2, ноутбук Samsung R580-JS03 Intel Core i5 430M 2.26 ГГц, DDR3-1066 3 Гб, Ubuntu 12.10;

экспериментальная платформа 3, ноутбук MacBook Air Intel Core i7 3667U 2 ГГц, DDR3L-1600 МГц 8 Гб, OS X 10.8.5.

Предварительные испытания показали, что наибольшей интенсивности генерации сообщений нельзя достигнуть при использовании однопоточного приложения. Несмотря на то что экспериментальное приложение формировало около ста тысяч сообщений в секунду, реальное количество отправляемых сообщений было значительно меньше. По мере увеличения количества запущенных экземпляров экспериментальной программы количество отправляемых сообщений возрастало, поэтому в процессе эксперимента запускалось несколько экземпляров. Для определения их количества предварительно было измерено, сколько экземпляров программы можно запустить до того момента, когда потери пакетов при перехвате составят около 50 % (при помощи программы tcpdump). Исходя из этого, во время эксперимента выбирались такие условия, когда потери не превышают 5 %.

В ходе эксперимента было проведено более миллиона измерений на каждой из платформ. Для получения достоверных результатов эксперимент повторялся десять раз. Такое количество повторов было выбрано исходя из объема обрабатываемых данных и затрат времени. Например, один эксперимент занимает 1 мин 50 с, но при этом объем данных составляет 1,2 Гб для одной платформы.

Результаты измерения времени, необходимого экспериментальным платформам для обработки сообщения, приведены в табл. 1 и 2. На рис. 1–3 отражено изменение интенсивности генерации сообщений и среднего времени обработки одного сообщения для каждой платформы.

В табл. 1 приведены обобщенные результаты для всех испытаний. В столбцах $t_{\min}$ и $t_{\max}$ приведено наибольшее и наименьшее время обработки пакета, обнаруженное в ходе десяти испытаний. Детализированные результаты для каждого испытания отражены в табл. 2.

Погрешность среднего рассчитывалась с помощью следующей формулы:

$$S_x = \frac{S}{\sqrt{n}} = \sqrt{\sum_{i=0}^{n} \frac{(t_i - \bar{t})^2}{n(n-1)}}.$$

Для расчета погрешности среднего между испытаниями применялся коэффициент Стьюдента, равный 2,262 (число измерений 10, надежность 0,95).

На представленных графиках видно, что по мере увеличения интенсивности генерации сообщений среднее время их обработ-

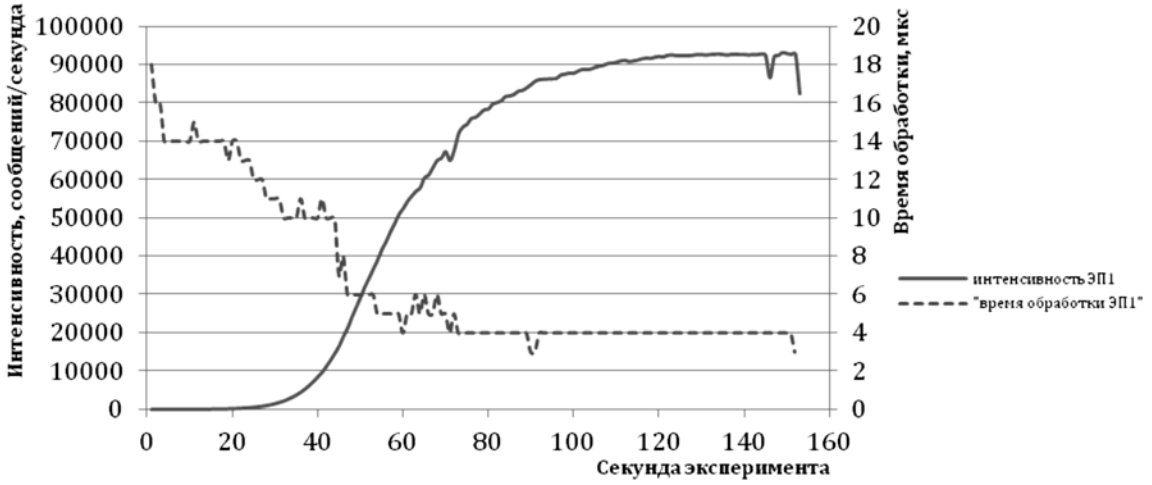

Рис. 1. Изменение интенсивности и среднего времени обработки ЭП1

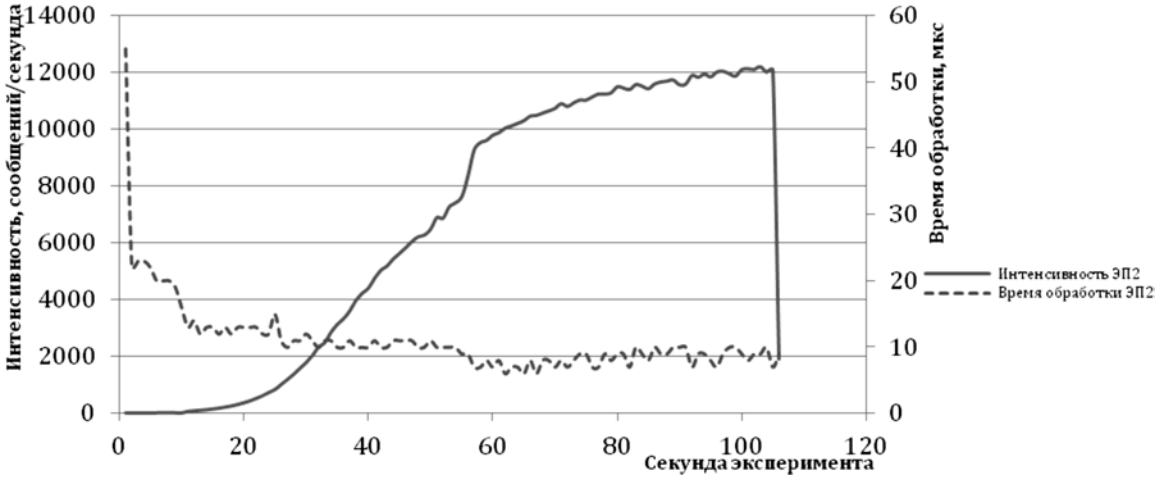

Рис. 2. Изменение интенсивности и среднего времени обработки ЭП2

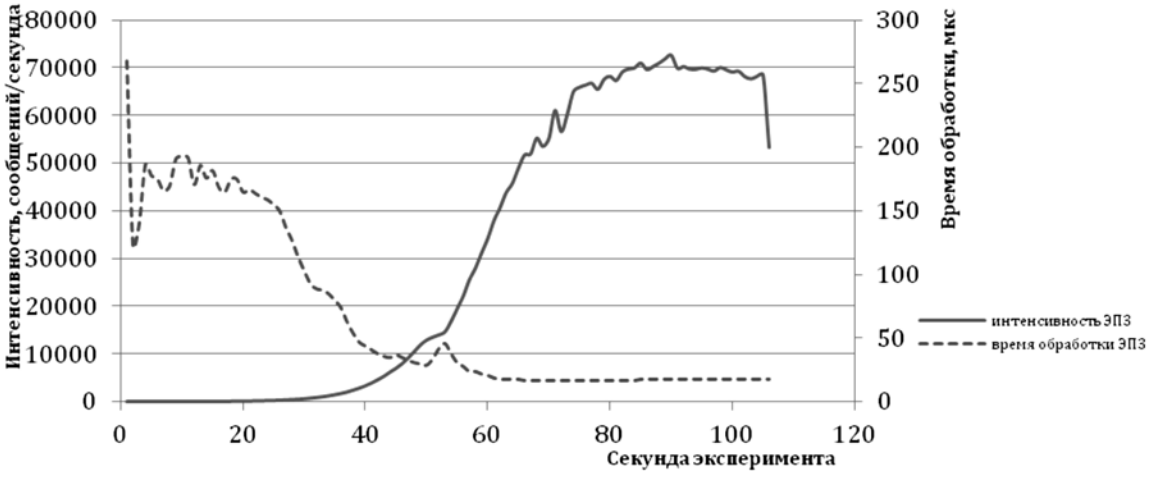

Рис. 3. Изменение интенсивности и среднего времени обработки ЭП3

Таблица 1

Обобщенные результаты измерений времени обработки

Платформа	Количество испытаний	$t_{среднее}$, мкс	t_{min}, мкс	t_{max}, мкс	Погрешность среднего между испытаниями, мкс
Общие показатели					
ЭП 1	10	9,05	2	1704	± 1,6323
ЭП 2		10,35	2	3770	± 0,5810
ЭП 3		60,23	< 0	31740	± 0,9627
Предельные показатели					
ЭП 1	10	3,83	2	835	± 0,0721
ЭП 2		8,20	2	3770	± 0,0779
ЭП 3		17,97	< 0	2030	± 0,2056

ки уменьшается. По этой причине в табл. 1 приведены результаты измерений времени обработки в тот момент, когда интенсивность генерации достигала наибольшего значения. Кроме того, для более детального изучения этого процесса были проведены дополнительные эксперименты, но из-за ограниченного объема статьи здесь их результаты не отражены.

На время обработки сообщений влияет множество факторов: запущенные приложения, настройка и реализация сетевого стека, режимы управления энергопитанием компьютерного узла и т. д. С учетом этого эксперименты проводились в условиях, когда платформы были не нагружены другими задачами. И даже в таком случае, при проведении двух одинаковых экспериментов в одинаковых условиях, полученные результаты имели различия, хотя и незначительные, связанные с внутренними процессами ОС и компьютерной системы в целом.

Следующий эксперимент был посвящен проверке предположения, что поток получаемых из сети сообщений может влиять на нагружающие способности узла. В ходе проведения нагрузочного тестирования со стороны тестируемого узла формируется поток ответов на запросы нагружающего узла. Каждый полученный ответ обрабатывается операционной системой до того, как он будет проанализирован приложением. Этот поток ответов в нашей работе мы называем *обратной нагрузкой*. Кроме того, использование ряда протоколов сетевого стека приводит к тому, что взаимодействующие узлы обмениваются служебной информацией. Все это потребляет вычислительные ресурсы компьютера и может способствовать снижению предельной нагрузочной способности узла.

Для проверки данного предположения платформы были соединены в локальную сеть. Предварительные испытания показали, что экспериментальная платформа 3 лучше всего подходит на роль узла, показания которого будут считываться и анализироваться в ходе эксперимента. Это связано с тем, что один экземпляр экспериментальной программы на ЭП3 способен генерировать сообщения с большой интенсивностью при одновременном перехвате пакетов без потерь (в программе tcpdump).

Эксперимент состоял из следующих шагов:

- запуск перехватчика tcpdump на узлах ЭП1 и ЭП2, создающих обратную нагрузку;
- запуск экспериментальной программы на ЭП3;
- наблюдение за создаваемой нагрузкой с помощью программы мониторинга сетевой активности, встроенной в ОС ЭП3;
- при достижении ЭП3 пикового значения интенсивности генерации пакетов осуществлялся запуск программы tcpdump на ЭП3;
- запуск экспериментальной программы на ЭП1 и ЭП2;

Таблица 2

Детализированные результаты измерений времени обработки

Платформа	Номер испытания	Обработано пакетов	$t_{среднее}$, мкс	t_{min}, мкс	t_{max}, мкс	Погрешность измерений среднего, мкс
ЭП 1	1	4 636 952	10,52	2	900	± 0,00287
	2	4 623 601	8,06	2	257	± 0,00185
	3	4 607 858	8,39	2	194	± 0,00199
	4	4 631 816	7,95	2	101	± 0,00178
	5	4 674 155	15,16	2	1704	± 0,00491
	6	4 632 833	8,09	2	281	± 0,00192
	7	4 644 583	7,99	2	424	± 0,00191
	8	4 593 897	8,23	2	835	± 0,00202
	9	4 526 941	8,20	2	402	± 0,00196
	10	4 636 628	7,87	2	384	± 0,00185
ЭП 2	1	1 013 345	12,56	2	1817	± 0,03862
	2	1 013 754	9,89	2	2461	± 0,03932
	3	1 025 806	10,59	2	1838	± 0,03674
	4	1 023 268	9,84	2	1786	± 0,0357
	5	1 015 334	10,05	2	1818	± 0,0401
	6	1 029 289	9,83	2	1805	± 0,03882
	7	1 026 739	9,99	2	2123	± 0,03823
	8	1 027 369	10,17	2	3770	± 0,03811
	9	1 029 575	10,24	2	1799	± 0,0393
	10	1 027 199	10,38	2	1796	± 0,03648
ЭП 3	1	5 443 042	61,27	<0	1231	± 0,01816
	2	5 455 387	57,17	<0	447	± 0,0164
	3	5 432 302	60,44	<0	2031	± 0,01777
	4	5 434 095	61,44	<0	1290	± 0,01828
	5	5 454 387	60,99	<0	410	± 0,018
	6	5 486 672	61,51	<0	2090	± 0,0182
	7	5 598 231	59,84	<0	31740	± 0,01839
	8	5 608 325	59,63	<0	2049	± 0,0174
	9	5 617 287	59,14	<0	556	± 0,01721
	10	5 607 822	60,91	<0	458	± 0,01795

• наблюдение на ЭП3 за обратной нагрузкой с помощью программы мониторинга;

• при достижении пикового значения создаваемой обратной нагрузки осуществлялось завершение работы экспериментальной программы и перехватчика пакетов на узлах ЭП1 и ЭП2;

• завершение работы экспериментальной программы и перехватчика пакетов на ЭП3;

• сбор и анализ результатов.

Результаты проведенного эксперимента отражены на рис. 4.

На представленном графике можно видеть, что по мере увеличения обратной нагрузки происходит снижение создаваемой. После прекращения генерации обратной нагрузки интенсивность генерации пакетов исследуемой системы восстанавливается. Одновременно с этим увеличивается

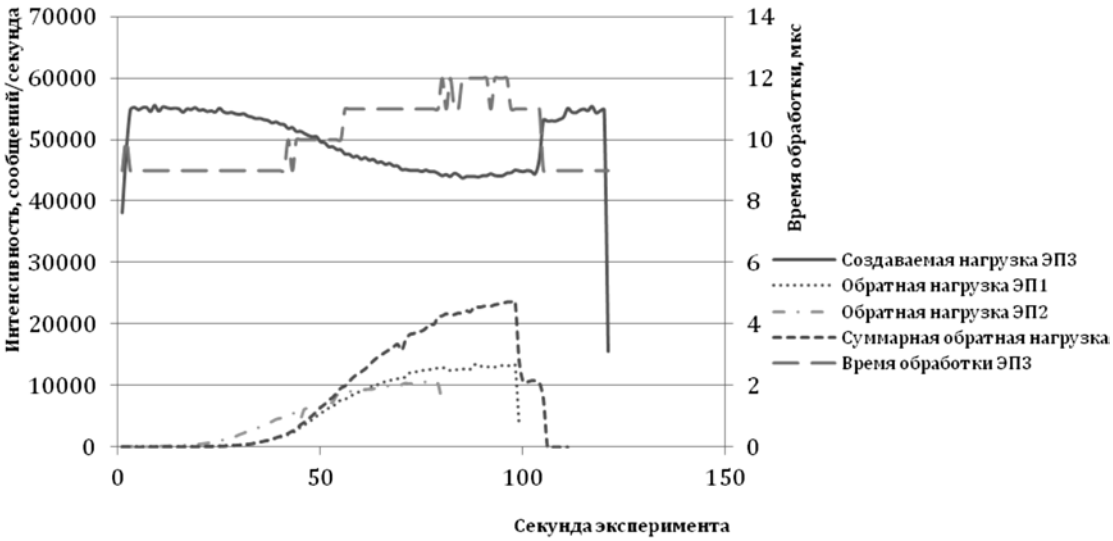

Рис. 4. Изменение интенсивности и среднего времени обработки в ходе эксперимента

среднее время обработки сообщения операционной системой. В нашем исследовании данную особенность мы назвали *эффектом обратной нагрузки*.

В описанных выше экспериментах использовались популярные современные операционные системы, за исключением Microsoft Windows. Это вызвано тем, что точность системного таймера такой ОС не позволяет измерить время обработки одного сообщения сетевым стеком, и ее измерения затруднительны.

Проведенные эксперименты позволяют оценить нагрузочные способности существующих компьютерных систем. При этом экспериментальная программа и перехватчик пакетов выступали в роли аналога модуля нагрузки и сборщика метрик производительности. Предельные нагрузочные способности каждой из экспериментальных платформ разительно отличаются друг от друга. Наибольшая интенсивность генерации сообщений, без потерь данных, составила около 90 000 в секунду на ЭП1. Используя предельные результаты измерений можно рассчитать теоретическую нагрузочную способность. Для ЭП1 она составляет 261 096, для ЭП2 – 121 951 и для ЭП3 – 55 648 сообщений в секунду. Как можно видеть, теоретическая нагрузочная способность ЭП3 меньше практической. По всей видимости, это связано с особенностями операционной системы данной платформы.

Результаты проведенных экспериментов демонстрируют характеристики операционных систем и компьютерных узлов, используемых в настоящее время для проведения нагрузочного тестирования. Полученные данные позволяют сделать вывод о том, что имеющая место нехватка нагрузочных способностей компьютерных узлов обусловлена как внутренними, так и внешними факторами. Модули инструментов нагрузочного тестирования, анализирующие ответы от тестируемого узла, вместе с ресурсоемкими транспортными протоколами и эффектом обратной нагрузки снижают нагрузочные способности компьютера. Это свидетельствует о том, что применение компьютеров со стандартной архитектурой для решения задач нагрузочного тестирования неэффективно.

Учитывая полученные результаты, нами был разработан метод проведения нагрузочного тестирования, основанный на применении аппаратного модуля с программируемой логической интегральной схемой (ПЛИС). С его помощью может осуществляться создание нагрузки на тестируемую систему и сбор результатов тестирования. Результаты проведенной работы вместе с характеристиками модуля будут опубликованы в ближайшее время.

СПИСОК ЛИТЕРАТУРЫ

1. **Meier J.D., Farre C., Bansode P., Barber Sc., Rea D.** Performance Testing Guidance for Web Applications [электронный ресурс] / URL: http://msdn.microsoft.com/en-us/library/bb924375.aspx (дата обращения: 10.03.2014).

2. **Силаков Д.В.** Автоматизация тестирования web-приложений, основанных на скриптовых языках // Труды ИСП РАН. 2008. № 2. С. 159–178.

3. **Сортов А., Хорошилов А.** Функциональное тестирование web-приложений на основе технологии UniTesK // Труды ИСП РАН. 2004. № 8. С. 77–97.

4. **Яковенко П.Н., Сапожников А.В.** Инфраструктура тестирования веб-сервисов на базе технологии TTCN-3 и платформы.NET // Труды ИСП РАН. 2009. № 117. С. 63–74.

5. **Zhang P., Elbaum S., Dwyer M.B.** Automatic Generation of Load Tests // Proc.of the 26th IEEE / ACM Internat. Conf. on Automated Software Engineering. 2011. Pp. 43–52.

6. **Zhang P., Elbaum S., Dwyer M.B.** Compositional load test generation for software pipelines // Proc. of Internat. Symp. on Software Testing and Analysis. 2012. Pp. 89–99.

7. **Morrison K., Haddad H.M.** Converting users to testers: an alternative approach to load test script creation, parameterization and data correlation // J. of Computing Sciences in Colleges. 2012. Vol. 28. Iss. 2. Pp. 188–196.

8. **CaiYu., Grundy J., Hosking J.** Synthesizing client load models for performance engineering via web crawling // Proc. of the 22nd IEEE/ACM Internat. Conf. on Automated Software Engineering. 2007. Pp. 353–362.

9. **Ермыкин А.А.** Разработка метода построения комплекса нагрузочного тестирования распределенной информационной системы: Дис. ... канд. техн. наук. СПб., 2005.

10. **Zhen Ming Jiang.** Automated analysis of load testing results // Proc. of the 19th Internat. Symp. on Software Testing and Analysis. 2010. Pp. 143–146.

11. **Haroon Malik.** A Methodology to Support Load Test Analysis // Proc. of the 32nd ACM/IEEE Internat. Conf. on Software Engineering. 2010. Vol. 2. Pp. 421–424.

12. Top 15 Performance Testing Tools [электронный ресурс] / URL: http://www.toolsjournal.com/tools-world/item/156-top-performance-testing-tools (дата обращения 10.03.2014).

13. Performancetesttools [электронный ресурс] / URL: http://www.opensourcetesting.org/performance.php (дата обращения 10.03.2014).

14. **Serpanos D., Wolf T.** Architecture of Network Systems. Burlington: Morgan Kaufmann Publishers, 2011. 313 p.

15. Pcap [электронный ресурс] / URL: http://ru.wikipedia.org/wiki/Pcap (дата обращения 10.03.2014).

REFERENCES

1. **Meier J.D., Farre C., Bansode Pr.,Barber Sc., Rea D.** Performance Testing Guidance for Web Applications, *Microsoft patterns & practices,* 2007, Available: http://msdn.microsoft.com/en-us/library/bb924375.aspx (Accessed 10.03.2014).

2. **Silakov D.V.** Avtomatizatsiya testirovaniya web-prilozheniy, osnovannykh na skriptovykh yazykakh,*Trudy Instituta sistemnogo programmirovaniya RAN*, 2008, No. 2, Pp. 159–178. (rus)

3. **Sortov A., Khoroshilov A.** Funktsionalnoye testirovaniye web-prilozheniy na osnove tekhnologii UniTesK,*Trudy Instituta sistemnogo programmirovaniya RAN*, 2004, No. 8, Pp. 77–97. (rus)

4. **Yakovenko P.N., Sapozhnikov A.V.** Infrastruktura testirovaniya web-servisov na baze tekhnologii TTCN-3 i platformy.NET, *Trudy Instituta sistemnogo programmirovaniya RAN*, 2009, No. 117, Pp. 63–74. (rus)

5. **Zhang P., Elbaum S., Dwyer M.B.** Automatic Generation of Load Tests, *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering,* 2011, Pp. 43–52.

6. **Zhang P., Elbaum S., Dwyer M.B.** Compositional load test generation for software pipelines,*Proceedings of the International Symposium on Software Testing and Analysis,* 2012, Pp. 89–99.

7. **Morrison K., Haddad H.M.** Converting users to testers: an alternative approach to load test script creation, parameterization and data correlation, *Journal of Computing Sciences in Colleges,* 2012, Vol. 28, Iss. 2, Pp. 188–196.

8. **Cai Yu., Grundy J., Hosking J.** Synthesizing client load models for performance engineering via web crawling, *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering,*2007, Pp. 353–362.

9. **Ермыкин А.А.** *Razrabotka metoda postroyeniya kompleksa nagruzochnogo testirovaniya raspredelennoy informatsionnoy sistemy*: dis. ... kand. tekhn. nauk, St. Petersburg, 2005. (rus)

10. **Zhen Ming Jiang.** Automated analysis of load testing results, *Proceedings of the 19th International Symposium on Software Testing and Analysis,* 2010, Pp. 143–146.

11. **Haroon Malik.** A Methodology to Support Load Test Analysis, *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering,* 2010, Vol. 2, Pp. 421–424.

12. *Top 15 Performance Testing Tools.* Available: http://www.toolsjournal.com/tools-world/item/156-top-performance-testing-tools (Accessed 10.03.2014).

13. *Performance test tools.* Available: http://www.opensourcetesting.org/performance.php (Accessed 10.03.2014).

14. **Serpanos D., Wolf T.** Architecture of Network Systems. Burlington: Morgan Kaufmann Publishers, 2011, 313 p.

15. *Pcap.* Available: http://ru.wikipedia.org/wiki/Pcap (Accessed 10.03.2014).

БОРОДИН Антон Александрович – *ассистент Московского государственного университета леса.*
141005, Московская обл., г. Мытищи-5, ул. 1-я Институтская, д. 1.
E-mail: AntonioBorodin@gmail.com

BORODIN, Anton A. *Moscow State Forest University.*
141005, 1st Institutskaya Str. 1, Mytischi, Moscow region, Russia.
E-mail: AntonioBorodin@gmail.com