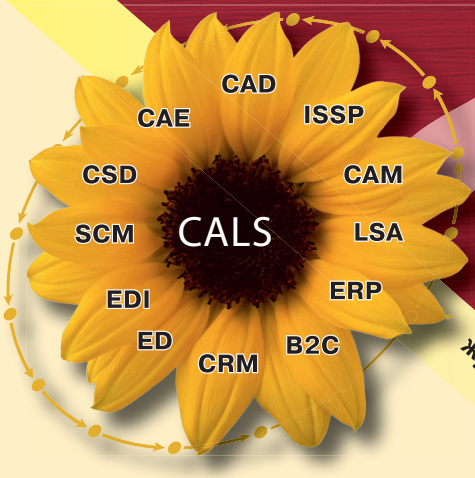
КАЧЕСТВО
ИННОВАЦИИ
ОБРАЗОВАНИЕ

№10
2014

ISSN 1999-513X



журнал в журнале

КАЧЕСТВО и ИПИ (CALS)-технологии

Качество • Инновации • Образование • №10-2014

www.quality-journal.ru

ЕЖЕМЕСЯЧНЫЙ НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

КАЧЕСТВО ИННОВАЦИИ ОБРАЗОВАНИЕ

№10 (113)
октябрь 2014

АДРЕС РЕДАКЦИИ И ИЗДАТЕЛЯ
105118, Москва, ул. Буракова, д. 8
Тел.: +7 (495) 916-89-29
Факс: +7 (495) 916-81-54
E-mail: quality@eqc.org.ru (для статей)
hg@eqc.org.ru (по общим вопросам)
www.quality-journal.ru; www.quality21.ru

ИЗДАТЕЛЬ
Европейский центр по качеству

НАУЧНЫЙ РЕДАКТОР
Гудков Ю.И.
ygudkov@hse.ru

ХУДОЖЕСТВЕННЫЙ РЕДАКТОР
Кудрявцева А.И.

ЛИТЕРАТУРНЫЙ РЕДАКТОР
Савин Е.С.

ОТВЕТСТВЕННЫЙ СЕКРЕТАРЬ
Мартюкова Е.С.
ne@eqc.org.ru

ЖУРНАЛ ЗАРЕГИСТРИРОВАН
в Министерстве РФ по делам печати, телерадиовещания и средств массовых коммуникаций. Свидетельство о регистрации ПИ №77-9092

ПОДПИСНОЙ ИНДЕКС
в каталоге агентства «Роспечать» 80620, 80621
в каталоге агентства «Урал-Пресс» 14490
на сайте НЭБ eLIBRARY.RU 80620

ОТПЕЧАТАНО
ФГУП Издательство «Известия» УД ПРФ
127254, г. Москва, ул. Добролюбова, д. 6

Журнал входит в перечень ВАК РФ

Статьи рецензируются

СОДЕРЖАНИЕ

Сведения о членах редколлегии и об авторах статей можно найти на сайте www.quality-journal.ru

Ю.Л. Леохин, И.Н. Дворецкий, А.С. Мягков

ОТЕЧЕСТВЕННАЯ ОПЕРАЦИОННАЯ СИСТЕМА CLOUD/IX ДЛЯ СЕРВЕРОВ НА ПРОЦЕССОРАХ АРХИТЕКТУРЫ ARM

В статье рассмотрены основные проблемы, связанные с развитием центров обработки и хранения данных, пути их решения, представлена отечественная операционная система Cloud/IX для серверов на процессорах архитектуры ARM, приведены результаты испытаний http-сервера nginx.

Ключевые слова: центр обработки и хранения данных, энергоэффективный серверный комплекс, операционная система, http-сервер, процессор

В настоящее время проблемы развития рынка центров обработки и хранения данных (ЦОХД) связаны с ростом потребления электроэнергии [1]. Затраты на электроэнергию составляют примерно половину от общих затрат владельцев ЦОХД на обслуживание серверного парка. Основные затраты на электроэнергию - это энергопотребление размещаемых в ЦОХД серверов и их охлаждение. Существуют несколько вариантов решения этой проблемы:

- эффективное использование существующих мощностей (например, использование технологий виртуализации и облачных вычислений [2], позволяющих повысить коэффициент использования существующего оборудования и в разы сократить потребности в оборудовании);
- новые решения архитектуры серверов и центров обработки и хранения данных (например, Cisco Unified Computing System [3]).

Одним из альтернативных и эффективных способов решения задачи снижения потребления электроэнергии является разработка серверного оборудования на базе энергоэффективных процессоров архитектуры ARM [4, 5]. По прогнозам IDC к 2015 году архитектура ARM может завоевать уже 15% рынка серверного оборудования [6]. Этот подход был использован в рамках реализации проекта «Разработка и организация высокотехнологичного производства энергоэффективных многопроцессорных аппаратно-программных серверных комплексов для государственных и корпоративных информационных систем и центров обработки данных», который выполняется сотрудниками Национального исследовательского университета «Высшая школа экономики».

Другая задача, которая решается в ходе реализации проекта – это увеличение эффективности использования вычислительных ресурсов центров обработки данных. Известно, что вычислительные ресурсы серверов в центрах обработки данных используются в среднем только на 20%. Использование технологий виртуализации и динамического выделения вычислительных ресурсов позволят существенно увеличить коэффициент полезного действия системы.

Третьей важной задачей является обеспечение масштабируемости системы, т.е. возможности постепенного наращивания вычислительных мощностей центра обработки данных за счет добавления новых модулей.

Четвертой задачей является повышение эффективности используемых площадей центра обработки данных. Используя новые подходы к построению архитектуры ЦОХД, можно будет добиться 2-3-кратного уменьшения площади, занимаемой серверной платформой при сохранении производительности.

В настоящее время рынок серверного оборудования на базе процессоров архитектуры ARM находится в начале своего развития. В сентябре 2010 г. британская компания ARM Holdings впервые вышла на серверный рынок, представив процессор Cortex-A15 на базе архитектуры ARMv7. Его версия ARM Cortex A15 MPCore, рассчитанная на тактовую частоту до 2,5 ГГц, в пять раз превосходила по производительности процессоры, применяемые в смартфонах, а по уровню энергопотребления была сопоставима с ними [6].

До недавнего времени рост рынка серверов на базе ARM сдерживался двумя факторами:

- отсутствием 64-разрядной версии процессора ARM;
- практически полным отсутствием серверного программного обеспечения под ARM.

Первая проблема была решена: недавно ARM Holdings выпустила первые 64-разрядные процессоры – Cortex-A57 и A53. Особые надежды связываются с 64-разрядной архитектурой ARM – ARMv8, которые планируется массово производить в 2014 году.

Для решения второй проблемы требуется создание новых программных продуктов, включая серверные ОС и средства виртуализации. В качестве примера способа решения этой задачи можно привести разработку транслятора двоичного кода x86 в ARM Российской компании «Эльбрус Технологии» [7], позволяющую переносить ПО для платформы x86 на серверы с процессорами архитектуры ARM. Программный эмулятор позволит без изменений исполнять на серверах ARM приложения, скомпилированные для архитектуры x86. Быстродействие транслированного кода составляет 45% от написанного специально для ARM. Планируется довести этот показатель до 80%.

В настоящее время ARM опубликовала пакет дополнений к ядру Linux, обеспечивающих поддержку набора инструкций ядра ARMv8. Эти дополнения реализованы в ряде дистрибутивов Linux, включая Ubuntu.

Таким образом, создание операционной системы для серверов на базе процессоров архитектуры ARM продолжает оставаться актуальной задачей, качественное решение которой позволит создавать максимально эффективные с точки зрения производительности, энергопотребления и масштабируемости серверные комплексы. **Поэтому разработка российской операционной системы Cloud/IX является важнейшей задачей, которая решается в рамках комплексного проекта, упомянутого выше.**

Коллектив разработчиков принял решение – с целью минимизации временных затрат на выполнение научно-исследовательских и опытно-конструкторских работ не начинать разработку ОС с «чистого листа», а выбрать в качестве прототипа уже существующую операционную систему. При выборе операционной среды для разрабатываемого в рамках комплексного проекта многопроцессорного аппаратно-программного серверного комплекса учитывалось множество критериев, среди которых можно выделить следующие:

- поддержка распределённых операций;
- масштабирование;
- лицензионная чистота;
- простота портирования драйверов и приложений;
- простота развёртывания и поддержки;
- стандартные интерфейсы;
- минимум оверхэда (overhead) для системных сервисов.

С учетом приведенных критериев был обоснован выбор в качестве прототипа операционной системы 9front [8, 9], на базе которой реализуется разработка отечественной проприетарной системы Cloud/IX.

9front и её исходный код распространяются по лицензии Lucent Public License, являющейся свободной лицензией. В частности, при бесплатном доступе к системе и исходному коду, разрешается коммерческое использование системы и её производных. При этом, в отличие от Gnu Public License, лицензия Lucent не требует “Copyleft”, т.е. изменения, внесённые в систему, остаются собственностью и производственным секретом разработчика и не подлежат обязательной публикации. Соответственно, лицензионные требования Lucent распространяются на все производные 9front, в частности, на Cloud/IX.

Главным преимуществом системы 9front является протокол ixP, позволяющий управлять локальными и распределёнными ресурсами путём простых отображений в пространстве имён. Наряду с множеством преимуществ, выбор нестандартной «экзотической» системы как основы системного ПО для продукта, несомненно, обладает и слабыми сторонами. Пожалуй, самой заметной из них является отличие набора системных интерфейсов от традиционного для подобных продуктов стандарта POSIX. Решение этой проблемы осуществляется по двум направлениям. Во-

первых, для 9front разработан пакет APE (ANSI/POSIX Environment), максимально приближающий системные интерфейсы к стандарту POSIX.

Во-вторых, принимая во внимание всё возрастающую популярность операционной системы Linux, разработчиками, совместно с компанией AltLinux, осуществляется портирование ОС Linux на целевую платформу (ARM, microTCA). Это позволит легко адаптировать для целевой платформы множество приложений, созданных для ОС Linux, включая традиционные кластерные приложения. Необходимо отметить, что многие полезные свойства дизайна 9front были адаптированы к Linux. Это касается, в частности, протокола ixP, использование которого в Linux теперь возможно на уровне монтируемых файловых систем, что позволяет осуществлять обмен файлами между Linux и 9front.

Так как Cloud/IX разработана на основе 9front, то она наследовала все особенности своего прототипа. Система построена на трех принципах:

- ресурсы именуются и доступны как файлы в иерархической файловой системе;
- стандартный протокол ixP для доступа к локальным и удаленным ресурсам;
- несвязанные иерархии, обеспечиваемые различными службами, соединяются вместе в единое собственное иерархическое пространство имен файлов.

Протокол ixP реализует множество транзакций, каждая из которых посылает запрос от процесса клиента локальному или удаленному серверу и возвращает результат. ixP контролирует файловую систему, а не только файлы. Доступ к файлам происходит на уровне байтов, а не блоков, что отличает ixP от таких протоколов, как NFS и RFS /10/.

В настоящее время разработана β-версия операционной системы Cloud/IX и выполняются работы по портированию наиболее популярных и часто используемых программных приложений, к которым относится nginx (nginx – веб-сервер и почтовый прокси-сервер, работающий на Unix -подобных операционных системах).

С целью исследования стабильности работы портированного http-сервера nginx на гетерогенной Cloud/IX, исследования возможности масштабирования и выявления дальнейших направлений работы, на базе Центра Обработки и Хранения Данных (ЦОХД) ООО “Системы и Решения” проводились испытания на программном макете в рамках комплексного проекта.

Во время испытаний в качестве аппаратной части использовались 24 вычислительные машины (блейд-серверы) с архитектурой x86, организованные в 3 системных крейта по 8 блейд-серверов в каждом.

Все блейд-серверы оборудованы как минимум одним контроллером Ethernet со скоростью 1000 Mbps.

При этом проводился мониторинг загрузки узлов кластера средствами ОС (отдельно по каждой подсистеме) в процессе нагрузочного тестирования, позволяющий сделать предварительные выводы о путях потенциального улучшения производительности. Для отображения результатов мониторинга была разработана программа сбора статистики с нескольких узлов кластера, её объединения на одном узле, перевода результатов мониторинга в формат, удобный для анализа и отображения в реальном масштабе времени.

Программа получает данные о загрузке узлов и сетевых интерфейсов и отображает их в наглядном виде на веб-странице.

Решение реализовано при использовании следующего технологического стека:

- серверная часть приложения написана на языке clojure – реализации lisp-диалекта для JVM и библиотек: Ring, Compojure, Webbit, Clj-json;
- клиентская часть реализована на языке ClojureScript – диалекте Clojure, транслирующемуся в обычный Java Script, выполняемый в браузере;
- реализация отправки сообщений с сервера на клиент базируется на технологии вебсокетов (WebSockets);
- динамическая отрисовка графических элементов реализована через работу с элементом Canvas (спецификация HTML5).

В таблице 1 приведен пример кода для визуализации загрузки кластера.

Таблица 1. Пример кода для визуализации загрузки кластера

```
(ns monitor.monitor)
(def vertices [[0 0] [250 0] [0 250] [250 250]])
(def vertices-for-connections [[100 100] [250 100] [100 250] [250 250]])
(def router-vertices [[145 145] [205 145] [145 205] [205 205]])
(defn vertex-x [vertex-number] ((vertices vertex-number) 0))
(defn vertex-y [vertex-number] ((vertices vertex-number) 1))
(def conn-count (atom 0))
(defn draw-node-line [context vertex-number number style]
 (let [x (vertex-x vertex-number)
       y (vertex-y vertex-number)
       h (* number 10)]
 (set! (.-fillStyle context) style)
 (set! (.-fillStyle context) style)
 (.fillRect context x (+ y h) 100 9)))
(defn draw-connector-line [context x1 y1 x2 y2 style]
  (. context (beginPath))
  (set! (.-strokeStyle context) style)
  (set! (.-lineWidth context) 4)
  (.moveTo context x1 y1)
  (.lineTo context x2 y2)
  (. context (stroke)))
(defn draw-connector [context vertex-number style]
 (let [x1 ((vertices-for-connections vertex-number) 0)
       y1 ((vertices-for-connections vertex-number) 1)
       x2 ((router-vertices vertex-number) 0)
       y2 ((router-vertices vertex-number) 1)]
   (draw-connector-line context x1 y1 x2 y2 style)))
(defn draw-node-line-free [context vertex number]
 (draw-node-line context vertex number "#000"))
(defn draw-node-line-loaded [context vertex number]
 (draw-node-line context vertex number "red"))
(defn draw-basic-node [context vertex]
  (doseq [x (range 10)]
     (draw-node-line-free context vertex x)))
(defn draw-all [context]
(do
 (doseq [x (range 4)]
     (do
       (draw-basic-node context x)
       (draw-connector context x "#ccc")))
(.fillRect context 145 145 60 60)))
(defn load-node [context vertex percent]
 (let [num (int (/ percent 10.0))]
    (doseq [x (range num)]
       (draw-node-line-loaded context vertex (- 10 x)))
    (doseq [x (range num 10)]
      (draw-node-line-free context vertex (- 10 x)))))
(def conn  (js/WebSocket. "ws://127.0.0.1:8081/ws"))
(set! (.-onopen conn)
(fn [e]
  (.send conn
         (.stringify js/JSON (js-obj "msg" "connect")))))
(set! (.-onerror conn)
(fn []
  (js/alert "error")
  (.log js/console js/arguments)))
(defn send-to-server [msg]
(.send conn  (.stringify js/JSON (js-obj "msg" msg))))
(defn load-node-next [context vertex percent]
```

Для генерации запросов к балансировщику нагрузки nginx используется httpperf. Httperf - это утилита измерения производительности веб-сервера, предоставляющая гибкие условия для генерации рабочих нагрузок на HTTP-сервер и измерения его производительности. По окончании работы он формирует отчет, в котором содержатся три раздела: общие результаты, группа, посвященная соединениям и группа.

В таблице 2 приведен пример пустого отчета утилиты httperf.

Таблица 2. Пример пустого отчета утилиты httperf

```
Total: connections 1000 requests 0 replies 0 test-duration 0.106 s
Connection rate: 9444.1 conn/s (0.1 ms/conn, <=1 concurrent connections)
Connection time [ms]: min 0.0 avg 0.0 max 0.0 median 0.0 stddev 0.0
Connection time [ms]: connect 0.1
Connection length [replies/conn]: 0.000
Request rate: 0.0 req/s (0.0 ms/req)
Request size [B]: 0.0
Reply rate [replies/s]: min 0.0 avg 0.0 max 0.0 stddev 0.0 (0 samples)
Reply time [ms]: response 0.0 transfer 0.0
Reply size [B]: header 0.0 content 0.0 footer 0.0 (total 0.0)
Reply status: 1xx=0 2xx=0 3xx=0 4xx=0 5xx=0
CPU time [s]: user 0.02 system 0.08 (user 22.5% system 77.5% total 100.0%)
Net I/O: 0.0 KB/s (0.0*10^6 bps)
Errors: total 1000 client-timo 0 socket-timo 0 connrefused 1000 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

В режиме генерации нагрузки он формирует запросы с подстановкой возрастающего числа, цифры которого используются в качестве компонентов пути к ресурсу.

Сценарий тестирования эффективности балансировки нагрузки требует создания на каждом узле с nginx иерархии файлов так, чтобы их пути относительно корневого каталога nginx соответствовали запросам, формируемым httpperf, и при этом общий объём данных превышал бы размер оперативной памяти каждого узла. При этих условиях узким местом на каждом узле становится дисковая подсистема, благодаря чему становится возможным оценить эффект от распараллеливания нагрузки.

Одна и та же последовательность неповторяющихся запросов подается на балансировщик и на отдельный узел, и результаты сопоставляются.

Производительность отдельных узлов и всего кластера также тестируется с помощью повторяющихся (идентичных) запросов. httpperf позволяет регулировать количество запросов в единицу времени, что отражается на количестве параллельно обрабатываемых запросов. Тест проводится отдельно для загрузки большого файла и загрузки небольшого файла, благодаря чему можно выявить различные потенциальные узкие места в портированном nginx. В этом сценарии тестирования всё содержимое файла находится в кэше ОС, и дисковая подсистема более не является узким местом.

Для того чтобы результаты тестирования отражали производительность серверного решения (nginx), в отличие от всего комплекса клиент-сервер, требуется либо обеспечить наличие множества компьютеров-клиентов, генерирующих запросы одновременно, либо использовать для запуска клиента систему, существенно превосходящую по производительности совокупность всех узлов кластера (без учёта дисковой подсистемы, которая не используется клиентом активно). На текущий момент было выбрано второе решение.

Основной показатель, характеризующий результат тестирования - количество обрабатываемых запросов в секунду по итоговой статистике httpperf. Другой показатель, имеющий большое значение при содержательном анализе нагрузки и узких мест - частота запросов в единицу времени, при которой сервер перестаёт успешно обрабатывать все соединения (для его получения проводится несколько запусков httpperf, где опция -rate последовательно увеличивается, при этом проверяется статистика успешных запросов).

Мониторинг загрузки узлов кластера средствами ОС (отдельно по каждой подсистеме) в процессе нагрузочного тестирования также позволяет сделать предварительные выводы о путях потенциального улучшения производительности (рисунок 1).

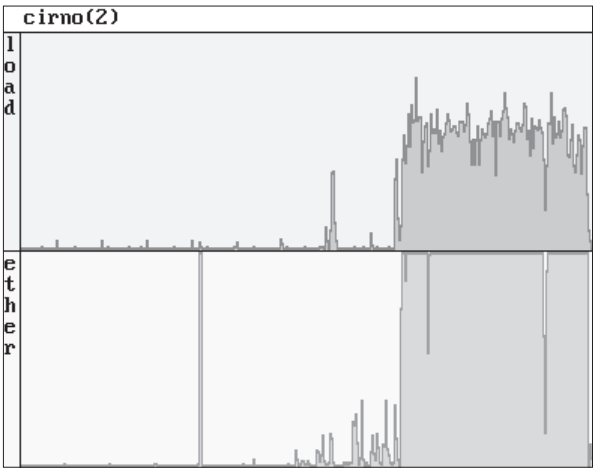


Рис. 1. Загрузка процессора и сетевого адаптера ключевого узла при проведении тестирования

В ходе тестирования было проведено четыре серии экспериментов. В ходе каждого опрашивался кластер из блейдов количеством от 4 до 24 (таблица 3).

Таблица 3. Результаты экспериментов по нагрузочному тестированию

Серия экспериментов	Соединений в секунду, среднее	Время эксперимента, с, среднее	Частота отклика, мин, соед./с	Частота отклика, сред, соед./с	Частота отклика, макс, соед./с	СКО, %
серия 1	128,0	1022.6	74,4	128,0	196,8	7.4
серия 2	324,0	404.9	177,6	321,6	477,6	9.8
серия 3	416,0	315,3	258,2	414,4	592,0	11,4
серия 4	1054,6	125,9	750,0	1051,6	1591,2	16,8

Таблица показывает масштабируемость серверов nginx, запущенных на операционной системе Cloud/IX в ЦОХД.

Тем не менее, в ходе экспериментов возник ряд проблем, связанных с соединениями между серверами nginx под управлением Cloud/IX. При перегрузке серверов запросами nginx выбрасывал пакеты (connection refused), которые приходили в фазу обработки соединения другого пакета. График зависимости доли отказов для серии экспериментов 2 от частоты запросов показан на рисунке 2. Разработчики считают, что это связано с некорректной обработкой сокетов в системе Cloud/IX, а также с особенностями реализации TCP-стека в дистрибутиве 9front, который был взят за основу. Изучив то, каким образом 9front обращается с протоколом TCP, разработчики выяснили, что перед SYN+ACK ядро 9front не проверяет, хватит ли клиенту места в очереди на обработку, и выкидывает RST уже после установления соединения, что не противоречит стандарту RFC 793, в котором не описано, каким образом сервер должен обрабатывать переполнение очереди: до или после ACK. Таким образом, к вышеупомянутой проблеме, связанной с работой со стеком TCP, можно относиться как к особенности Cloud/IX, которую следует учитывать при дальнейшей работе с программным обеспечением, использующим протокол TCP.

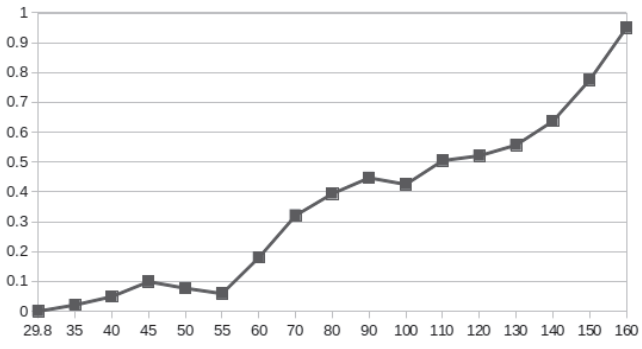


Рис. 2. График зависимости доли отказов для серии экспериментов 2 от частоты запросов, соед./100мс.

Кроме того, в данной операционной системе изначально не предусмотрено использование сокетов, и для их реализации были взяты портированные сокеты BSD, которые не учитывают архитектурные особенности сетевого стека Cloud/IX. Исходя из тестов, разработчики предполагают, что сетевая производительность может быть существенно увеличена, если использовать сокеты, учитывающие эти особенности. В будущем планируется реализация сокетов беркли поверх протокола 9p, использующегося в Cloud/IX.

В отличие от многих других операционных систем, Cloud/IX изначально не представляет специальных программных интерфейсов для доступа к устройствам. Вместо этого в системе предполагается контролировать взаимодействие через файловую систему, которая может быть предоставлена по протоколу 9p, в том числе и для устройств. Взаимодействие предполагается обычными операциями ввода/вывода. Этим фактом объясняется отсутствие сокетов в Cloud/IX, и, тем самым, ставится задача разработчикам для их реализации на операционной системе.

По итогам испытаний макета был установлено, что сетевые серверные приложения могут быть перенесены и использоваться на Cloud/IX, и показывать производительность, сопоставимую с аналогичными устройствами. Кроме того, выявлен ряд особенностей ядра 9front и поставлены задачи разработчикам по поиску способов работы с ними и их описанию.

Литература

1. ServerWatch: IT buying more x86 servers, paying less (IDC): http://www.serverwatch.com/stats/article.php/3884731/IT-Buying-More-x86-Servers-Paying-Less-IDC.htm

2. VMware Server Virtualization Products for Virtual Data Center Management https://www.vmware.com/products/datacenter-virtualization.html

3. Cisco Unified Computing System http://www.cisco.com/web/solutions/data_center/unifiedcomputing_promo.html

4. Леохин Ю.Л., Дворецкий И.Н. Тенденции развития науки и техники в области производства серверного оборудования для дата-центров // Изв. Вузов. Приборостроение. 2013. Т. 56. № 12. С. 20-24.

5. Леохин Ю.Л., Дворецкий И.Н., Салибекян С.М. Энергоэффективный многопроцессорный аппаратно-программный комплекс для ЦОД // Системный администратор. 2013. № 12 (133). С. 66-68.

6. Орлов С. Революция ARM // Журнал сетевых решений / LAN. №11. 2012. http://www.osp.ru/lan/2012/11/13032394/

7. http://servernews.ru/596643

8. Пайк Р., Пресотто Д., Доруорд С., Фландрена Б., Томпсон К., Трики Х., Уинтерботтом Ф. Операционная система Plan 9 от Bell Labs // Открытые системы. № 061 . 1995.

9. Пайк Р., Пресотто Д., Доруорд С., Фландрена Б., Томпсон К., Трики Х., Уинтерботтом Ф. Операционная система Plan 9 от Bell Labs // Открытые системы. № 01 . 1996.

10. Trikey H. APE - The ANSI/POSIX Environment, Plan 9 Programmer's Manual, V. 2, AT&T Bell Laboratories, Murray Hill, NJ, 1991.

Леохин Юрий Львович,
профессор НИУ ВШЭ
yleokhin@hse.ru

Дворецкий Игорь Николаевич,
научный сотрудник НИУ ВШЭ
idvoretskiy@hse.ru

Мягков Андрей Сергеевич,
аспирант НИУ ВШЭ
myagkov.as@gmail.com

Y.L. Leokhin, I.N. Dvoretskiy, A.S. Myagkov

PATRIOTIC OS CLOUD / IX SERVER PROCESSORS ARM

The article describes the main issues related to the development centers and storage, solutions, presented the domestic operating system Cloud / IX server processors architecture ARM, hello test results of http-server nginx.

Keywords: processing center and storage, energy-efficient server complex, the operating system, http-server, processor

Bibliography

1. ServerWatch: IT buying more x86 servers, paying less (IDC): http://www.serverwatch.com/stats/article.php/3884731/IT-Buying-More-x86-Servers-Paying-Less-IDC.htm

2. VMware Server Virtualization Products for Virtual Data Center Management https://www.vmware.com/products/datacenter-virtualization.html

3. Cisco Unified Computing System http://www.cisco.com/web/solutions/data_center/unifiedcomputing_promo.html

4. Leokhin Y.L., Dvoretskiy I.N. Trends in the Development of Science and Technology in the Sphere of the Servers Production for Data Centers // Izv. Vuzov. Priborostroenie. 2013. V. 56, № 12, PP. 20-24.

5. Leokhin Y.L., Dvoretskiy I.N., Salibekjan S.M. Power effective multiprocessor hardware-software complex for data-processing centers // System administrator. 2013. № 12 (133), PP. 66-68.

6. Orlov S. Revolution ARM // Journal of network solutions /LAN №11, 2012, http://www.osp.ru/lan/2012/11/13032394/

7. http://servernews.ru/596643

8. Pike R., Presotto D., Dorward S., Flandrena B., Thompson K., Trickey H., Winterbottom P. USENIX Association Plan 9 from Bell Labs // Open Systems, № 06 , 1995, PP. 57-64.

9. Pike R., Presotto D., Dorward S., Flandrena B., Thompson K., Trickey H., Winterbottom P. USENIX Association Plan 9 from Bell Labs // Open Systems, № 01 , 1996, PP. 70-77.

10. Trikey H. APE - The ANSI/POSIX Environment, Plan 9 Programmer's Manual, V. 2, AT&T Bell Laboratories, Murray Hill, NJ, 1991.

Y.L. Leokhin,
professor, HSE
yleokhin@hse.ru

I.N. Dvoretskiy,
researcher, HSE
idvoretskiy@hse.ru

A.S. Myagkov,
postgraduate student, HSE
myagkov.as@gmail.com