Открытые технологии

Материалы седьмой международной конференции
Linux Vacation /Eastern Europe 2011

ОТКРЫТЫЕ ТЕХНОЛОГИИ

Сборник материалов седьмой
международной конференции
разработчиков и пользователей
свободного программного обеспечения
Linux Vacation / Eastern Europe 2011

(г. Гродно, 30 июня–03 июля 2011 г.)

Брест
«Альтернатива»
2011

УДК 004.45(082)
ББК 32.973.26-018.2я43
О-83

Под общей редакцией кандидата технических наук,
доцента **Д.А. Костюка**

Рецензенты:
кандидат технических наук, доцент, заведующий кафедрой электронных вычислительных машин и систем Брестского государственного технического университета **С.С. Дереченник;**
кандидат технических наук, доцент кафедры ЭВМ Белорусского университета информатики и радиоэлектроники **Д.И. Самаль**

Открытые технологии : сборник материалов седьмой междунар. конф. разработчиков и пользователей свобод. програм. обеспечения Linux Vacation / Eastern Europe 2011 (Гродно, 30 июня - 03 июля 2011 г.) / под общей ред. Д.А. Костюка. – Брест : Альтернатива, 2011. – 132 с.

О-83

ISBN 978-985-521-251-6.

В сборник вошли стати, посвященные новым технологиям и разработкам в сфере свободного (открытого) программного обеспечения и затрагивающие широкий спектр платформ – от рабочих станций и серверов, включая решения на базе виртуализации, до встраиваемых систем и мобильных устройств. Тематические направления включают разработку и тестирование программного обеспечения, администрирование систем, построенных на основе свободных программ, правовые и организационные особенности использования свободных лицензий.

Сборник может быть интересен специалистам в области информационных технологий, занимающимся разработкой и сопровождением свободного программного обеспечения, а также подготовкой кадров высшей квалификации по профильным специальностям.

УДК 004.45(082)
ББК 32.973.26-018.2я43

ISBN 978-985-521-251-6

Содержание

Cells.js — ещё один подход к разработке современных веб-приложений

Алексей Кондратенко*

Membase NoSQL database management system has administrative interface implemented as modern 'single-page' aka 'pure-js' web application. This talk will describe Cells.js — a library that was grown inside this user interface component. It facilitates structuring of user interface as a collection of inter-dependent variables or cells. This provides some otherwise hard to implement features, like back button support, for free. And I believe, that this leads to cleaner and smaller code base.

В последние годы получила распостранение такая практика организации веб-приложений, когда на стороне браузера реализуется вся логика пользовательского интерфейса. Серверная сторона при этом предоставляет только «голый» API для доступа к данным. Преимуществом такого подхода является более отзывчивый интерфейс, т. к. реакция на действия пользователя реализуется на стороне браузера.

Одной из проблем построения пользовательского интерфейса в таком виде является относительная неадаптированность современных web-стандартов к такого рода приложениям. Эти стандарты создавались в основном для статического web. Другой известной проблемой является неполная или неправильная реализация стандартов в разных браузерах.

Для реализации несложной логики зачастую достаточно применить одну из библиотек для кросс-браузерного манипулирования DOM и кросс-браузерной реализации XMLHttpRequest. В последнее время в этой нише доминирует библиотека jquery. Реализация более сложной логики быстро превращает код в запутанную и трудно поддерживаемую кашу из обработчиков событий и требует перехода к какому-либо способу организации более богатого пользовательского интерфейса.

К одному из таких способов относятся MVC и его производные. На сегодняшний день имеется масса тяжеловесных фреймвор-

*Altoros Development, alk@tut.by

ков для реализации MVC, например Google Web Toolkit, ExtJS, SproutCore.

Однако, на мой взгляд, в средних по размеру приложениях применение более компактных и простых решений оправдано.

При создании web-интерфейса для Membase было изначально решено использовать только jquery. Однако после реализации первых экранов я понял, что без более продвинутой организации логики этот код будет невозможно поддерживать и развивать.

Большинство MVC-фреймворков имеет в своем основании систему наблюдения за данными и реакции на их изменения. Это позволяет отделять логику получения/обработки данных от логики обработки пользовательской реакции и логики представления этих данных пользователю.

Я начал с создания простого класса, реализующего наблюдаемое значение и возможность «подписаться» на его изменения. Это помогло структурировать код лучше. Но затем я заметил, что некоторые значения полностью детерминированно зависят от других значений. Я вспомнил про проект cells, реализованный на Common Lisp (`http://common-lisp.net/project/cells/`) и понял, что могу легко реализовать основную идею этого проекта на javascript.

В результате пользовательский интерфейс Membase (и до этого Northscale Memcached Server) организован как набор связаных зависимостями по данным ячеек. Большинство ячеек является вычисляемыми, т. е. каждой из них соответствует детерминированная функция на javascript. Значения функций вычисляемых ячеек могут зависеть только от значений других ячеек. И библиотека организует (пере)вычисления ячеек когда значения их зависимостей либо становится известно, либо изменяется.

Когда одной из исходных ячеек присваивается какое-либо значение, например, в результате действия пользователя, cells.js организует перевычисление всех ячеек, которые зависят от этой ячейки. После чего перевычисляются ячейки, зависимые от только что перевычисленных ячеек, и так далее. Т. е. cells.js организует «расплывание» данных (и, в некотором роде, реакции) по ячейкам.

Видимые пользователю блоки пользовательского интерфейса подписываются на значение ячейки, которую отображают, и обновляются автоматически.

Cells.js позволяет связывать hash-фрагменты URL с ячейками. Так что кнопка «назад», которая просто меняет URL страницы, работает автоматически. При этом просто меняется значение одной

из исходных ячеек, и остальная часть пользовательского интерфейса обновляется соответственно.

Cells.js реализует получение данных с сервера HTTP GET-запросом как один из видов вычисления значения. Пока запрос загружается, значение ячейки — undefined. Зависимые от этой ячейки блоки интерфейса обычно реагируют на это отображением индикатора загрузки. Это поведение реализуется автоматически в коде связывания ячейки и HTML-шаблона с блоком.

URL GET-запросов как правило берется из других ячеек. В результате REST API Membase большей частью реально реализован через гипертекст (что соответствует каноническому определению REST). Ссылки на GET-запросы берутся из ответов на другие запросы.

Изменение ячейки с URL побуждает зависимую ячейку с содержимым этого URL выполнить запрос повторно с новым URL.

Так, например, реакция пользователя может поменять имя (или какой-либо идентификатор) выделенного обьекта. Зависимая от имени ячейка может находить URL этого обьекта по имени в «листинге» обьектов, который содержит список пар имя — URL, и ячейка с описанием этого обьекта может просто получать его GET-запросом с сервера. Смена выделенного имени будет автоматически приводить к получению актуального описания выделенного обьекта, и отображение этой ячейки будет также меняться автоматически.

Считаю, что такой способ организации пользовательского интерфейса является очень логичным и продуктивным. Он позволил в очень короткие сроки реализовать довольно продвинутый web-интерфейс Membase.

На момент написания этого текста cells все еще является частью Membase. Как и весь код Membase, код cells доступен по лицензии Apache версии 2.0. В ближайшее время планирую закончить работу по вынесению cells в отдельный независимый free software проект.

Организация программной защиты от DDoS-атак

Олег Бойцев*

DDoS Mitigation Software Solutions. The report outlines experience in deploying software protection against DDoS attacks using OS Linux. The following issues of organization of DDoS attacks are reviewed: the psychology of the attackers, the tools and techniques they use. In the report of the conference participants will receive an answer to the question: Is it possible to defeat DDoS without hardware tools and if so, how?

Введение

Тема DDoS/antiDDoS бесспорно является интересной и актуальной.

Для многих компаний предоставление сервисов и услуг online является чуть ли не единственной формой ведения бизнеса (`www.mastercard.com`, `www.facebook.com`, `www.habrahabr.ru`) в связи с чем, доступность сервиса онлайн можно считать критически важным условием успешного развития компании. Простой онлайн сервиса, даже на не продолжительное время, для компании может оказаться слишком дорогим «удовольствием»...

Экономика DDoS-атак

Как и в реальной экономике, расчет стоимости DDoS-атаки на сайт определяется прежде всего соотношением спрос/предложение. Цена также определяется и «весом» сайта — чем крупнее клиент, тем больше обычно просят за него. Для примера: на момент написания тезисов доклада средняя стоимость ддос атаки на среднестатистический сайт составляла $50. Приблизительно такова же и стоимость защиты в сутки. Стоимость покупки ботнета составляет $1500–$2000.

*Минск, [Mega-Admin.com][, `infosecurity@ya.ru`

Какие DDoS-атаки бывают, какие из них используют чаще всего

В зависимости от целей чаще всего используют следующие типы атак:

- HTTP flood,
- TCP flood,
- UDP/ICMP flood.

Инструменты для проведения DDoS-атак

За последние годы инструменты проведения DDoS-атак значительно видоизменились. То, что раньше было доступно узкому кругу избранным, в настоящее время поставлено на поток. Общей тенденцией развития инструментов проведения ддос атак можно считать: использование децентрализованной структуры управления ботнетом, упрощение user-end интерфейсов управления, шифрование трафика боты — командный центр.

Модель многоуровневой защиты от DDoS-атак

Firewall → Linux kernel → scripts → nginx → apache

- Firewall: ограничиваем количество соединений в единицу времени, режем ботов через string module
- Linux kernel: уменьшаем таймауты на обработку соединений, повышаем лимит общего количества возможных соединений, предел используемой памяти.
- Scripts: режем сетевые аномалии
- Nginx: тюнингуем, прикручиваем limit_conn
- Apache: тюнингуем, прикручиваем mod-evasive

Что можно победить

Все что не толще ширины канала сервера.

В случае если толщина DDoS-атаки больше ширина канала — железно фильтруем трафик на входе в ДЦ.

Шляхі акселерацыі выканання вылічальнай нагрузкі на Python

Антон Літвіненка*

Interpreted programming languages are known for their flexibility and convenience, but suffer from low execution speed comparing to compiled ones. However, they find a use for time-consuming applications. The enhanced methods of acceleration for interpreted languages are available, first of all just-in-time (JIT) compilation. The effect of the JIT compilation on the execution speed of a time-consuming scientific program is reported, and a few implementations of JIT compilers for Python are compared. *PyPy* was found to be the fastest one, *Psyco* — slightly slower, *Unladen Swallow* — nearly useless.

Інтэрпрэтаваныя мовы праграмавання ўсё часцей ужываюцца пры напісанні праграм для разнастайных сфераў дастасавання, у тым ліку такіх, што традыцыйна не асацыююцца з інтэрпрэтаванымі мовамі, а менавіта — напісанне праграм, якія патрабуюць вялікіх вылічальных рэсурсаў.

Так, напрыклад, існуе часткова напісаная на Python бібліятэка для квантавахімічнага мадэлявання *pDynamo* [1], аўтары якой адмовіліся ад традыцыйнага ў гэтай галіне выбару між Fortran, C або C++ і сваіх ранейшых напрацовак на Fortran, і аддалі перавагу інтэрфейсу на Python (з рэалізацыяй найбольш крытычнай да хуткасці часткі коду на C). Прычыны поспеху інтэрпрэтаваных моў у гэтай галіне ў выпадку навуковых праграм, верагодна, абумоўленыя лёгкасцю распрацоўкі для неадмыслоўцаў.

Таксама гэтыя мовы зручнейшыя для распрацоўкі з элементамі рэфлектыўнага праграмавання (калі праграме неабходна генераваць частку ўласнага коду).

Аўтарам дакладу была распрацаваная праграма *Mjöllnir*, якая выконвае інтэрпрэтацыю вынікаў магнетахімічнага эксперыменту з выкарыстаннем паўнаматрычнага метаду рашэння аператарных ураўненняў для спін-гамільтаніяна магнітных узаемадзеянняў, зыходзячы з мадэлі, якая задаецца карыстальнікам [2]. Для рашэння

*Кіеўскі нацыянальны ўніверсітэт імя Тараса Шаўчэнкі, tenebrosus.scriptor@gmail.com

гэтае задачы неабходна на падставе мадэлі пабудаваць матрыцы спін-гамільтаніяна (прыклад на рыс. 1), прычым у алгебраічным (сімвальным) выглядзе. Алгарытм пабудовы быў рэалізаваны на Python. Праграму можна атрымаць ад аўтара пад умовамі ліцэнзіі GNU GPL v3.

Безумоўна, праблема хуткасці выканання ў такім выпадку становіцца актуальнай.

z	$\lvert+\frac{1}{2};+\frac{1}{2}\rangle$	$\lvert+\frac{1}{2};-\frac{1}{2}\rangle$	$\lvert-\frac{1}{2};+\frac{1}{2}\rangle$	$\lvert-\frac{1}{2};-\frac{1}{2}\rangle$
$\langle+\frac{1}{2};+\frac{1}{2}\rvert$	$\mu H_z g_z - \frac{1}{2}J_z$	0	0	0
$\langle+\frac{1}{2};-\frac{1}{2}\rvert$	0	$\frac{1}{2}J_z$	$-J_z$	0
$\langle-\frac{1}{2};+\frac{1}{2}\rvert$	0	$-J_z$	$\frac{1}{2}J_z$	0
$\langle-\frac{1}{2};-\frac{1}{2}\rvert$	0	0	0	$-\mu H_z g_z - \frac{1}{2}J_z$

Рыс. 1. Матрыца спін-гамільтаніяна для асі z для комплексу Cu_2.

Метады акселерацыі выканання прадугледжваюць адыход ад чыстай інтэрпрэтацыі зыходнага коду ў момант выканання і пераход да разнастайных гібрыдных схемаў трансляцыі.

Па-першае, самая крытычная да рэсурсаў частка функцыянальнасці можа быць напісаная на кампіляванай мове праграмавання. Але такі падыход патрабуе перапісвання коду і можа ўскладняць рэалізацыю.

Ускосным дастасаваннем такога падыходу з'яўляецца выкарыстанне адных канструкцый мовы замест іншых, аналагічных, але хутчэйшых[3]. Так, сапраўды, інструкцыя:

```
map(operator.add, l1, l2)
```

хутчэй, чым: `map(lambda x,y: x+y, l1, l2)`. А спроба сабраць матрыцу ў адзін радок інструкцыяй `collect_str+=a[i][j]` у цыкле істотна прайграе інструкцыі:

```
"".join(map(lambda x:"".join(x),a)).
```

У той жа час, неабходна дакладна ведаць спосабы рэалізацыі інструкцый (якія могуць змяняцца з часам і залежаць ад рэалізацыі).

Прэкампіляцыя ў байт-код з'яўляецца традыцыйнай і выкарыстоўваецца заўсёды, калі ёсць такая магчымасць.

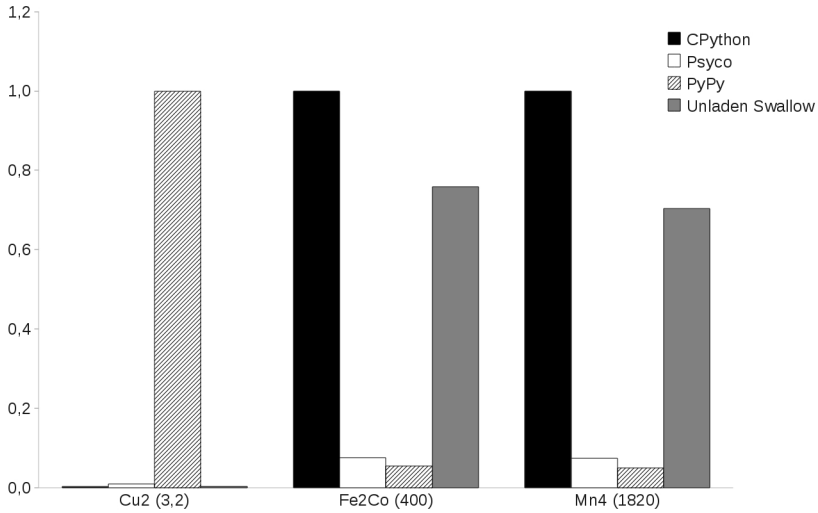
Існуюць праекты статычных кампілятараў падмноства мовы Python у машынны код, напрыклад, *Shedskin*. Праект *PyPy*, акрамя іншых дастасаванняў, здольны статычна кампіляваць г.зв. падмноства RPython. На жаль, гэты падыход абмяжоўвае магчымасці мовы (у прыватнасці, яго складана ўжыць для ўжо гатовага коду).

Найбольш цікавымі з'яўляюцца праекты JIT-кампілятараў:

- Модуль *Psyco*. Рэалізаваны як модуль *CPython* (стандартнае рэалізацыі мовы). З'яўляецца найбольш старым, вядомым, зручным, а да апошняга часу — і самым хуткім. Недахопы: прывязаны да версіі інтэрпрэтатара і платформы (толькі x86), прычым распрацоўка новых версій ускладненая (напрыклад, версіі пад Python 2.7 няма дагэтуль).
- Праект *PyPy*. Інтэпрэтатар і JIT-кампілятар. Хуткасць выканання расце ад версіі да версіі і ўжо перавышае хуткасць Psyco [4]. Ускладненая праца з вонкавымі бібліятэкамі на C.
- Праект *Unladen Swallow*. Пазіцыянуецца як аптымізаваны *CPython* з JIT-кампіляцыяй. Пакуль што працуе нашмат павольней за папярэднія два.

Агульная праблема ўсіх JIT-рэалізацый — патрабаванне большага аб'ёму памяці, чым *CPython*. Вынікі сінтэтычных тэстаў хуткасці наяўныя ў Сеціве [4]. Для праверкі эфекту акселерацыі ў выпадку праграмы *Mjöllnir* былі выкананыя тэсты для біядзернага комплексу медзі (Cu_2, 4 базісныя функцыі), трыядзернага Fe_2Co (432) і 4-ядзернага Mn_4 (625). Усярэдненыя вынікі для 10 запускаў (Core 2 Duo 2.4 GHz, Linux Fedora 12) прадстаўленыя на рыс. 2 і ў табл. 1, давяральныя інтэрвалы былі разлічаныя паводле размеркавання Ст'юдэнта.

Такім чынам, для праграмы *Mjöllnir* найбольш эфектыўным акселератарам выканання на дадзены момант з'яўляецца *PyPy*, і з улікам імклівага развіцця праекта верагодным ёсць ягонае выкарыстанне ў будучыні як асноўнага (цяпер ужываецца *Psyco*). *Psyco* дае блізкія вынікі, *Unladen Swallow* з'яўляецца неэфектыўным. Неадпаведнасць вынікаў для малой мадэлі (Cu_2), верагодна, абумоўленая выдаткамі часу на JIT-кампіляцыю, якія не кампенсуюцца праз вельмі малы аб'ём разлікаў.



Рыс. 2. Дыяграма параўнання адносных часоў выканання праграмы *Mjöllnir* з рознымі JIT-кампілятарамі (значэнні нармаваныя на час самага павольнага выканання для мадэлі (у дужках, сек)).

Таблица 1. Час выканання праграмы *Mjöllnir*, сек.

Мадэль	Cu_2	Fe_2Co	Mn_4
Базісныя функцыі	4	432	625
CPython 2.6.2	$0{,}0075 \pm 0{,}0002$	400 ± 7	1820 ± 20
Psyco 1.6 / CPython 2.6.2	$0{,}0304 \pm 0{,}0002$	$30{,}5 \pm 0{,}3$	134 ± 1
PyPy 1.5	$3{,}2 \pm 0{,}1$	$21{,}8 \pm 0{,}6$	91 ± 2
Unladen Swallow 2009Q3	$0{,}010 \pm 0{,}005$	303 ± 4	1280 ± 20

Аўтар дзякуе С. А. Літвіненку за каштоўныя парады пры выкананні гэтае работы.

Літаратура

[1] M. J. Field. The pDynamo Library for Molecular Simulations using Hybrid Quantum Mechanical and Molecular Mechanical Potentials, J. Chem. Theo. Comp., 4, 2008, 1151–1161.

[2] А. С. Литвиненко, Е. А. Михалева, С. В. Колотилов, В. В. Павлищук. Влияние спин-орбитального взаимодействия на магнитную восприимчивость полиядерных комплексов 3d-металлов, содержащих ион Co^{2+}, Теорет. и эксперим. химия, 2010, т. 46, с. 403–409.

[3] «Сказ о летающем змее. Агрессивная оптимизация программ на Python'е», http://www.xakep.ru/magazine/xa/123/102/1.asp

[4] «PyPy Speed Center: Comparison», http://speed.pypy.org/comparison/

Использование GNU Octave для инженерных и математических расчётов

Евгений Алексеев, Оксана Чеснокова*

Octave is a powerful program to solve engineering and mathematical problems. Its capabilities include packages for linear algebra, analytic geometry, mathematical analysis. Octave provides rather elaborated plotting. Octave can be used to solve differential equations, optimization problems and problems of processing the experiment data, occurring frequently in engineering and scientific practices.

Среди свободных математических программ наиболее известными являются Scilab и Maxima. Достойным конкурентом им может быть программа GNU Octave. Octave представляет собой интерактивный командный интерфейс для решения математических задач. Octave — это мощный математический язык интерпретирующего типа. Интерпретатор Octave входит в состав многих дистрибутивов Linux (Debian, Ubuntu, Mandriva, ALT Linux), есть версия и для ОС Windows.

Существуют и графические оболочки для работы с Octave (QtOctave, Xoctave), однако главным достоинством Octave является мощный математический язык интерпретирующего типа, а также наличие постоянно пополняющегося репозитария пакетов расширений octave-forge (`http://octave.sourceforge.net/`).

В Octave встроен язык программирования, очень близкий к языку проприетарной программы Matlab, позволяющий реализовать алгоритм любой сложности.

Функции Octave позволяют создавать графики и поверхности различной сложности. Для построения двумерных графиков можно использовать функции `plot` и `fplot`, полярные графики строятся с помощью встроенной функций `polar`, функции `mesh(x,y,z)`, `surf(x,y,z)` позволяют строить поверхности различного вида. По умолчанию для построения графиков и поверхностей в Octave используется пакет gnuplot (`http://www.gnuplot.info/`), который мо-

*Донецк, Украина, Донецкий национальный технический университет, EAlekseev@gmail.com

жет использоваться и как самостоятельная программа для построения графиков.

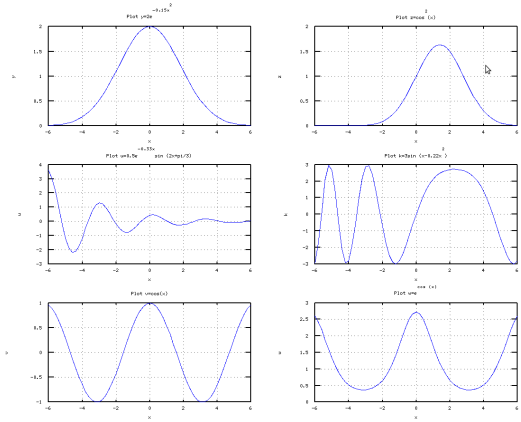


Рис. 3. Графики графики нескольких функций в одном окне

Octave содержит большое количество функций, предназначенных для решения задач линейной алгебры, наиболее используемые из них: `det(M)` — вычисляет определитель квадратной матрицы `M`, `norm(M[,p])` — возвращает различные виды норм матрицы `M` в зависимости от `p`, `inv(M)` — возвращает матрицу обратную к `M`, `eig(M)` — возвращает вектор собственных значений матрицы `M`, `rref(M)` — осуществляет приведение матрицы `M` к треугольной форме, используя метод исключения Гаусса, `lu(M)`, `qr (M)` — выполняют LU и QR-разложение соответственно.

Мощная графическая и математическая база Octave позволяет решать задачи векторной алгебры и аналитической геометрии.

Octave содержит функции для численного и аналитического решения нелинейных уравнений и систем, а также для интегрирования и дифференцирования.

Оптимизационные задачи чаще всего решают с помощью проприетарного табличного процессора MS Excel. Однако, наиболее мощные и гибкие функции для решения подобных задач присутствуют именно в Octave. Так, для решения линейных и нелинейных оптимизационных задач с ограничениями можно использовать функцию sqp, а для решения любых задач линейного программи-

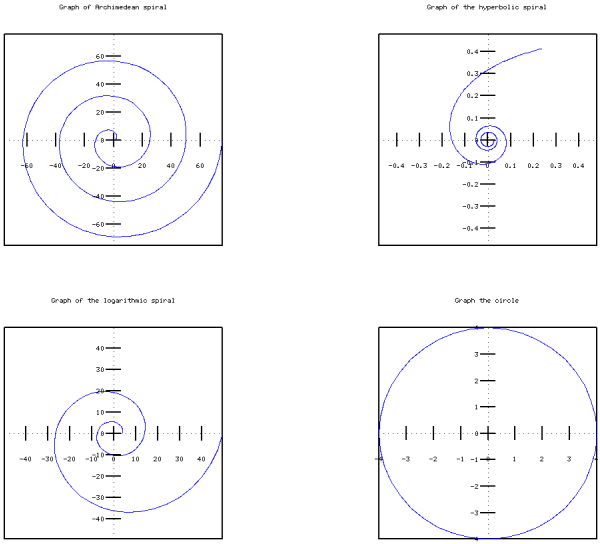


Рис. 4. Графики архимедовой, гиперболической и логарифмической спирали, окружности в полярных координатах

рования есть функция glpk. Сложные оптимизационные задачи в Octave решают с помощью пакета пакета расширений Minimization для GNU Octave (`http://octave.sourceforge.net/optim/overview.html`).

В инженерной практике часто приходится сталкиваться с решением обыкновенных дифференциальных уравнений и систем. В Octave существует достаточно много функций для решения обыкновенных уравнений и систем (в том числе и жёстких). Наиболее часто используемые среди них:

- `ode23`, `ode45` — функции решений обыкновенных нежёстких дифференциальных уравнений (или систем) методом Рунге-Кутта 2-3-го и 4-5-го порядка точности соответсвенно,
- `ode5r`, `ode2r` — функции решений обыкновенных жёстких дифференциальных уравнений (или систем).

Множество функций для решения дифференциальных уравнений находится в пакете расширений odepkg. Краткое описание функций этого пакета на английском языке с некоторыми примерами

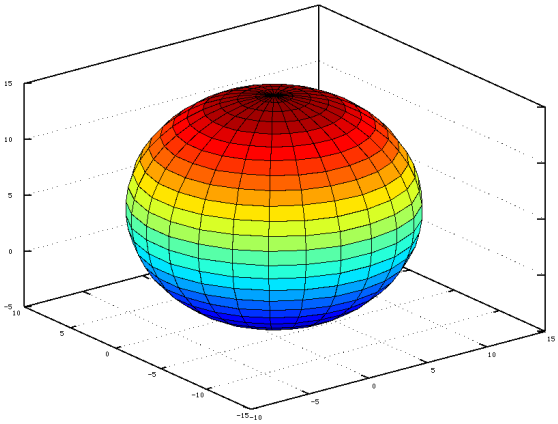


Рис. 5. График сферы

приведено на странице `http://octave.sourceforge.net/odepkg/overview.html`.

Также в Octave можно решать практически любые задачи обработки эксперимента. Для подбора параметров аналитической зависимости методом наименьших квадратов используются следующие функции: `polyfit` — подбор коэффициентов полинома k-й степени; `sqp` — функция поиска минимума.

Сплайн-интерполяция в Octave реализуется с помощью функции `interp1`, которая позволяет построить кубический и линейный сплайн.

С помощью Octave можно решать и много других задач. Авторами подготовлен учебник по использованию пакете GNU Octave, который будет опубликован в ближайшее время в Москве, в серии учебников ALT Linux, а также небольшим тиражом в Донецком национальном техническом университете. Рабочие материалы книги можно увидеть на сайте `http://gnu-octave.narod2.ru`.

Рассмотренные возможности Octave позволяют авторам рекомендовать пакет как инструмент для решения математических задач в курсах «Высшая математика», «Математический анализ», «Линейная алгебра», «Аналитическая геометрия», а также во мно-

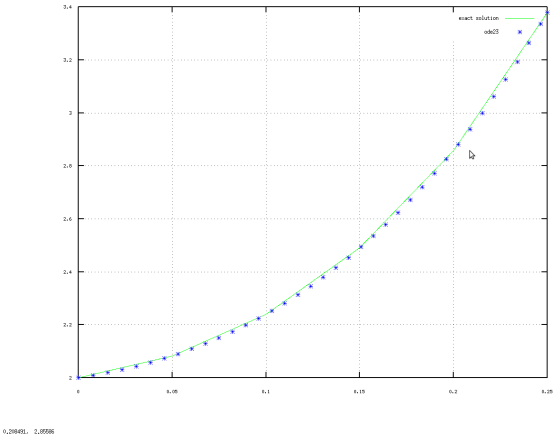


Рис. 6. Графики точного решения (–) дифференциального уравнений и решения (*), найденного с помощью ode23

гих специальных курсах, в которых приходится решать задачи вычислительной математики.

Выкарыстанне свабоднага праграмнага забеспячэння ва ўстановах адукацыі Украіны: спроба аналізу

Грыгорый Злобін*

The analysis of free / open source software usage in higher educational institutions of Ukraine is presented, based of the 'FOSS Lviv-2011' International Scientific Conference reports.

Нягледзячы на станоўчы досвед выкарыстання свабодных праграм у адукацыі як у краінах блізкага, так і далёкага замежжа, ва Украіне да гэтага часу не прынятая канцэпцыя выкарыстання свабоднага праграмнага забеспячэння у адукацыі. Разам з тым намаганнямі энтузіястаў у навучальных установах Украіны свабоднае праграмнае забеспячэнне ўсё ж выкарыстоўваецца! Праз адасобленую пазіцыю Міністэрства адукацыі і навукі Украіны няма падрабязнай інфармацыі аб досведзе выкарыстання свабодных праграм у адукацыі. Дзякуючы таму, што ў Львоўскім нацыянальным універсітэце імя Івана Франка 01–06 лютага 2011 адбылася даволі прадстаўнічая міжнародная навукова-практычная канферэнцыя «FOSS Lviv-2011», з'явілася магчымасць правядзення аналізу выкарыстання СПЗ у вышэйшых навучальных установах Украіны. З 81 дакладаў 49 было прысвечана выкарыстанню СПЗ у навучальных установах. Даклады [1], якія былі пададзеныя на гэтую канферэнцыю, можна згрупаваць па наступных кірунках (назва дакладу падаецца на мове арыгіналу):

Дыстанцыйнае навучанне

Гэтай тэматыцы прысвечаная найбольшая колькасць дакладаў, 10:

– «Розроблення електронного деканату для системи управління дистанційним навчання MOODLE» — Артеменко В.Б., Львоўская камерцыйная акадэмія

*Львоўскі нацыянальны ўніверсітэт ім. Івана Франка, zlobin@electronics.wups.lviv.ua

– «Вибір платформи дистанційного навчання» — Коцаренко М.В., Бойко О.В., Львоўскі нацыянальны медыцынскі ўніверсітэт ім. Данііла Галіцкага
– «Використання контрольно-діагностичної програми iTest у ході моніторингу якості процесу навчання старшокласників» — Макаренко І.Є., Мерзлікін П.В., Крыварожскі дзяржаўны педагагічны ўніверсітэт
– «Використання системи Moodle для організації контролю знань майбутніми вчителями-гуманітаріями» — Маркова Є.С., Бердзянскі дзяржаўны педагагічны ўніверсітэт
– «Тестування в Moodle як елемент менеджменту якості освіти: перший досвід» — Сергієнко В.П., Сліпухіна І.А., НПУ ім. М.П. Драгаманава
– «Особливості програмного забезпечення в електронному навчанні» — Жарких Ю.С., Лисоченко С.В., Сусь Б.Б., Третяк О.В., Кіеўскі нацыянальны ўніверсітэт ім. Тараса Шаўчэнкі
– «Інформаційно-аналітична система управління навчальним процесом ВНЗ на базі Moodle» — Триус Ю.В., Чаркаскі дзяржаўны тэхналагічны ўніверсітэт
– «Використання CMS JOMLA! та LCMS MOODLE у ВНЗ» — Франчук В.М., НПУ ім. М.П. Драгаманава
– «Локалізація системи MOODLE — Франчук В.М., НПУ ім. М.П. Драгаманава
– «Застосування вільного програмного забезпечення для дистанційного навчання у вищих навчальних закладах» — Захарченко В.М., Шапо В.М., Адэская нацыянальная марская акадэмія

Выкарыстанне сістэм кампутарнай матэматыкі

6 дакладаў можна аднесці да матэматычнай тэматыкі:

– «Використання вільно-поширюваного ПЗ математичного призначення в університеті» — Бугаєць Н.О., НПУ ім. М.П. Драгаманава
– «Вільнопоширювані системи комп'ютерної математики в освіті і науці» — Лазурчак І.І., Кобильник Т.П., Драгобыцкі дзяржаўны ўніверсітэт ім. І. Франка
– «Використання комп'ютерних математичних систем у професійній підготовці майбутнього вчителя математики» — Лов'янова І.В., Крыварожскі дзяржаўны педагагічны ўніверсітэт
– «Моделювання задач електротехніки у XCOS» — Філь І.М., Данецкі нацыянальны тэхнічны ўніверсітэт
– «Розробка і використання web-інтерфейсів для роботи з системами комп'ютерної математики» — Чичкарьов Є.А., Прыазоўскі дзяржаўны тэхнічны ўніверсітэт
– «Про комп'ютерний супровід викладання геометрії» — Яхненко І.В., Лутфулін М.В., Палтаўскі нацыянальны педагагічны ўніверсітэт ім. В.Г. Караленкі

Агульныя пытанні выкарыстання СПЗ у адукацыі

7 дакладаў другой па колькасці групы уключалі:

– «Використання вільного програмного забезпечення в навчанні і наукових дослідженнях у Львівському національному університеті імені Івана Франка» — Апуневич С.Є., Злобін Г.Г., Рикалюк Р.Є., Шувар Р.Я., ЛНУ ім. І. Франка
– «Використання вільного програмного забезпечення в системі дистанційної освіти» — Воронкін О.С., Луганскі дзяржаўны інстытут культуры і мастацтваў
– «Вільнопоширюване програмне забезпечення курсу "Нові інформаційні технології" для студентів спеціальності "Біологія"» — Єфименко В.В., НПУ ім. М.П. Драгаманава
– «Використання вільного програмного забезпечення у професійній підготовці майбутніх інженерів» — Покришень Д.А., Дрозд О.П., Сподаренко І.Й., Чарнігаўскі дзяржаўны тэхналагічны ўніверсітэт
– «Про досвід використання ОС Linux у навчальному процесі Львівського національного медичного університету імені Да-

нила Галицького» — Риковський П.А., Львоўскі нацыянальны медыцынскі ўніверсітэт ім. Данііла Галіцкага
- «LINUX та VIRTUAL-BOX у навчанні абстрактних понять теорії операційних систем» — Спірін О.М., Сверчевська О.С., Жытомірскі дзяржаўны ўніверсітэт ім. І. Франка
- «З досвіду використання вільного програмного забезпечення при вивченні інформатики» — Харченко В.М., Нежынскі дзяржаўны ўніверсітэт ім. М. Гогаля

Выкарыстанне адкрытых сродкаў праграмавання

Гэтая група апынулася найменшай па колькасці (4 даклады):
- «Використання відкритих програмних засобів в процесі навчання статистичним дисциплінам» — Коркуна Т.Й., Самбарскі тэхнікум эканомікі і інфарматыкі
- «Розрахунок фотоіонізаційних моделей небулярного газу в ОС LINUX UBUNTU 10.10 та WINDOWS 7» — Мелех Б.Я., Тишко Н.Л., Коритко Р.І., ЛНУ ім. І. Франка
- «Реалізація розподілених обчислень на основі грід-платформи з відкритим кодом BOINC» — Шийка Ю.Я., Шувар Р.Я., ЛНУ ім. І. Франка
- «Реалізація високопродуктивної обчислювальної системи на ОС LINUX» — Шувар Р.Я., Бойко Я.В., ЛНУ ім. І. Франка.

Распрацоўка праграмнага забеспячэння

Будучы тэматычна-роднаснай папярэдняй, гэтая група больш шматлікая і складаецца з 7 дакладаў, яна зраўнавалася з катэгорыяй «Агульныя пытанні выкарыстання СПЗ у адукацыі»:
- «Комплекс програм для лазерних спостережень штучних супутників Землі» — Мартинюк-Лотоцький К.П., Білінський А. І., ЛНУ ім. І. Франка
- «Розробка системи спектральної діагностики димової плазми» — Сподарець Д.В., Драган Г.С., Адэскі нацыянальны ўніверсітэт імя І.І. Мечнікава
- «Використання вільного програмного забез-печення для створення програми керування інформаційним автоматом» — Злобін Г., Скляр В., Чмихало О., Шевчик В., ЛНУ ім. І. Франка
- «Використання бібліотеки класів GEANT4 в ОС Linux при розробці програмного забезпечення для моделювання процесів

взаємодії випромінювання з речовиною» — Малихіна Т.В., Харкаўскі нацыянальны ўніверсітэт ім. В.Н. Каразіна
- «Програма для обробкирезультатів ЛЛС-спостережень як ВПЗ» — Апуневич С.Є., Апуневич С.В., Білінський А.І., Благодир Я.Т., ЛНУ ім. І. Франка
- «Програмне забезпечення керування телескопом ЛЛС-станції Львів-1831» — Білінський А.І., Мартинюк-Лотоцький К. П., ЛНУ ім. І. Франка
- «Використання Open Wrt як основи вбудованого програмного забезпечення маршрутизаторів» — Нек Т., ЛНУ ім. І. Франка

Асобныя даклады

І нарэшце, 5 дакладаў немагчыма аднесці ні да адной з пералічаных вышэй катэгорый:

- «Інформаційна технологія управління навчальним навантаженням у вищих навчальних закладах» — Гриценко В.Г., Чаркаскі нацыянальны ўніверсітэт ім. Б. Хмяльніцкага
- «Про досвід використання офісного пакету OpenOffice.org.ukr в курсі інформатики для економічних і юридичних спеціальностей ВЗО» — Злобін Г.Г., ЛНУ ім. І. Франка
- «Досвід використання редактора Gimp при вивченні курсу “Комп’ютерна графіка і дизайн”» — Матвієнко Ю.С., Палтаўскі нацыянальны педагагічны ўніверсітэт ім. В.Г. Караленкі
- «Використання програми GANTTPROJECT для побудови календарних графіків при розробці ПВР» — Грицук Ю.В., Мелiхов О.І., Данбаская акадэмія будаўніцтва і архітэктуры
- «Вільне ПЗ для підготовки наукових текстів і презентацій» — Лутфуллін М.В., Моторний М.І., Палтаўскі нацыянальны педагагічны ўніверсітэт ім. В.Г. Караленкі

Можна канстатаваць як шырокі спектр выкарыстання СПЗ ва ўкраінскіх навучальных установах ад дыстанцыйнага навучання да распрацоўкі адкрытага праграмнага забеспячэння, так і шырокую геаграфію выкарыстання СПЗ ад Луганска на ўсходзе да Львова на захадзе і ад Чарнігава на поўначы да Адэсы на поўдні.

Літаратура

[1] Тези міжнародної науково-практичної конференції FOSS LVIV-2011. Збірник наук. праць. ЛНУ ім. І. Франка, 2011. — 196 с.

Облачные вычисления и сервисы: классификация, основные функции, преимущества и недостатки

Виталий Сороко*

In the past, big part of computer-related activity was impossible without the installation of software to a local computer, but now, with help of cloud services, users can perform such typical tasks as word processing within their web-browser. It becomes possible due to so called cloud-based resources. Here an attempt of cloud technologies and systems general overview is presented with the technical description of the architecture, characteristics and features comparison.

Среди наиболее полных определений облачных систем на сегодняшний день можно выделить следующие два:

- Облачные сервисы — это технология обработки данных, в которой программное обеспечение предоставляется пользователю как интернет-сервис, при котором от пользователя скрыта инфраструктура «облака» (облачной системы) и, поэтому, ему не требуются специальные знания и навыки для управления и использования данной «облачной» технологии.
- Облачные вычисления — это вычисления, которые представляют собой динамически масштабируемый способ доступа к внешним вычислительным ресурсам в виде сервиса, предоставляемого посредством Интернета.

Данные определения тесно связаны между собой: для реализации всех облачных сервисов необходимы вычисления, а облачные вычисления по сути сами являются облачным сервисом. Рассмотрим классификацию облачных сервисов и их реализаций из мира свободного ПО.

В настоящее время все облачные сервисы подразделяют на:

- *«Программное обеспечение как услуга»* (Software as a Service, сокращённо SaaS) — бизнес-модель продажи программного

*Минск, http://arneta.ru

обеспечения, при которой владелец (поставщик) ПО предоставляет доступ к к нему пользователям (заказчикам) через Интернет. Примерами такого ПО являются Feng Office Community Edition, Simple Groupware, Zarafa и др.

– *«Оборудование (вычислительные мощности) как услуга»* (Hardware as a Service, сокращённо HaaS) — предоставление вычислительных ресурсов оборудования (его процессорного времени, места для место под хранения данных и т.д.) в виде сервисов с использованием технологий виртуализации. Сервисы обычно предлагаются как эквивалент реальным вычислительным системам, таким как серверы, суперкомпьютеры и др. Над программной реализацией этой идеи полностью или частично работают проекты OpenVZ, FreeVPS, Linux-VServer, Apache Hama, GlusterFS Open Source Project, а также Moose File System (MooseFS) и др., а предоставляет такой сервис на базе OpenSource решений компания Linode и некоторые другие.
– *«Коммуникация как Сервис»* (Communications as a Service, сокр. CaaS) — построенное в облаке коммуникационное решение для предприятия, которое обеспечивает передачу речевого сигнала по сети Интернет или по любым другим IP-сетям (VoIP), обмен мгновенными сообщениями (IM), видеоконференции. Модель CaaS позволяет деловым клиентам выборочно разворачивать средства коммуникаций и услуг на оснований оплаты услуг в срок для используемых сервисов. С этим направлением тесно связаны такие FOSS-проекты как Ekiga, iLBC, Speex.
– *«Мониторинг как Сервис»* (Monitoring-as-a-Service, сокращённо MaaS) является обслуживаемым в облаке программным обеспечением для мониторинга и обеспечения безопасности. Такими OpenSource-решениями на сегодняшний день являются Ganglia, Zabbix, Hyperic HQ. Сюда же с некоторыми оговорками модно отнести и Nagios.
– *«Инфраструктура как услуга»* (Infrastructure as a Service, сокращённо IaaS) — это предоставление компьютерной инфраструктуры (как правило в форме виртуализации) как услуги на основе концепции облачных вычислений. По сути IaaS является комбинацией SaaS, HaaS, так как она включает в себя и то и другое, причем обычно во множественном числе, а также CaaS и иногда MaaS с целью объедения и мониторинга всей

системы, и, поэтому, используется в основном предприятиями. Свободными реализациями данной концепции являются Eucalyptus, OpenNebula, OpenStack, Nimbus и др.

– *«Платформа как услуга»* (Platform as a Service, сокр. PaaS) — предоставление программной платформы и инструментов с определенными характеристиками, необходимых для разработки, тестирования, развертывания, поддержки различных приложений. Сюда же входят и готовые к использованию облачные сервисы, которые вместе образуют программную платформу. Яркими примерами из мира OpenSource в настоящее время являются Xen Cloud Platform, Cloud Foundry, Apache Hadoop, Apache Hive и др.
– *«Компьютер (виртуальный рабочий стол) как услуга»* (Desktop as a Service, сокращённо DaaS) — предоставление виртуального компьютера, который каждый пользователь может индивидуально настраивать под свои задачи. Таким образом, пользователь приходя на работу просто вводит свои данные (обычно логин и пароль) и может работать, используя при этом благодаря технологиям виртуализации вычислительные мощности стороннего сервера, а не своего ПК.
– *«Рабочее окружение как услуга»* (Workspace as a Service, сокращённо WaaS) — предоставление комплекта SaaS, предназначенного для создания рабочего окружения. В отличие от DaaS в этом случае пользователь получает доступ только к ПО, в то время как все вычисления происходят непосредственно на его машине. По сути данная категория является гибридом SaaS и PaaS, так как в отличие от последней является платформой, направленной не на разработку и тестирование ПО, а на офисную работу, но при этом как первая в реализации не использует технологий виртуализации. На данный момент реализации данной технологии предоставляются в основном различными крупными компаниями, например Google и Microsoft, и представляют в основном решения с закрытым исходным кодом, иногда с использованием свободных и открытых компонентов или их исходников.
– *«Все как услуга»* (Everything as a service, сокращённо EaaS) — концептуальная модель, включающая в себя элементы всех перечисленных решений. На данный момент полной её реализации не существует — она по сути является идеалом для крупных облачных компаний, таких как Google и Microsoft.

Свободное и открытое программное обеспечение в настоящее время играет ключевую роль в создании и развертывании облачных сервисов и систем. С одной стороны существуют ряд созданных сообществом платформ, ориентированных на облачные вычисления (Xen, Eucaliptus, Cloud Foundry, Feng Office и др.). С другой стороны, само свободное ПО (операционные системы семейства Linux и BSD, Web-браузеры и т. д.) как нельзя лучше подходит для размещения и использования облачных сервисов. Естественно, что существует и целый ряд проприетарных аналогов.

Рыночная доля облачных сервисов и платформ постоянно растет благодаря ряду преимуществ для обычных пользователей и организаций, среди которых в первую очередь можно отметить:

- количество процессоров, объем оперативной памяти и дискового пространства в облачных системах теоретически ничем не ограничен;
- пользователям не нужно самостоятельно устанавливать и настраивать ПО, т. к. для доступа к облачным сервисам достаточно обычного web-браузера;
- пользователям не нужно покупать дорогое оборудование;
- экономия времени и энергии на выполнение некоторых задач, а также, в особых случаях, и площадей, занимаемых оборудованием;
- возможность производить оплату только за потребленные вычислительные мощности и произведенные операции;
- в организациях отсутствуют затраты на развёртывание инфраструктуры;
- организации получают сокращение затрат на техническую поддержку и обновление развернутых систем, а также высокую скорость внедрения, обусловленную отсутствием временных затрат на развертывание системы;
- отсутствует необходимость обучения — большинство пользователей уже умеют пользоваться web-браузерами и интернет-сервисами как классом услуг;
- обычно облачные системы обслуживаются высококвалифицированными профессионалами, что дает более высокий уровень качества обслуживания ПО.

Тем не менее облачные системы не лишены недостатков, которые в большей степени касаются обычных пользователей, и в меньшей — провайдеров:

– из-за вопросов безопасности не все данные можно доверить стороннему провайдеру, тем более, не только для хранения, но и для обработки;
– далеко не каждое облачное приложение позволяет сохранить полученные результаты в удобном для пользователя виде на нужный носитель данных;
– риск массовой потери данных многими пользователями из-за технического сбоя у поставщика облачных услуг;
– потеря свободы — большая часть облачных сервисов не имеет четких стандартов, и поэтому при переходе от одного поставщика облачных услуг к другому могут возникнуть серьезные проблемы. Они же могут возникнуть и при обновлении провайдером собственных облачных сервисов — если, например, он пожелает внедрить новый интерфейс, то подписчикам придётся им пользоваться. Немаловажным моментом также является необходимость доступа в интернет, что в некоторых случаях ведет к потере свободы перемещений. А главное, благодаря тому, что все данные находятся в руках провайдера, нельзя исключать того, что недобросовестные компании могут воспользоваться этим.

Можно предположить, что как сейчас большая часть пользователей использует Windows и Microsoft Office, так в ближайшем будущем эти пользователи оценят преимущества облачных платформ и перейдут на них. При этом, даже передумав после получения очередной порции счетов за оплату сервисов, пользователям будет трудно вернуться к прежней схеме работы — все их данные будут в руках компании-владельца облачной системы, а для самостоятельной установки другой операционной системы и иного программного обеспечения многим не хватит квалификации, да и приобретённое пользователями оборудование скорее всего будет иметь крайне малую мощность. В этом случае эффективным выходом оказывается свободное ПО, которое как правило способно работать на маломощном оборудовании при сохранении совместимости со старыми системами. В этом случае возврат пользователей из облака также едва ли станет массовым, поскольку крупные облачные компании постараются сделать этот переход невыгодным, если не невозможным, однако сам факт такой возможности будет являться дополнительным регулирующим фактором, ограничивающим прибыли провайдеров облачных услуг.

Система распределенного выполнения задач paexec

Алексей Чеусов*

paexec distributes tasks across several CPUs or machines in a network and collects results from those CPUs/machines. paexec runs several instances of 'calculator' on remote/local machines and sends them tasks one by one recieving results from them. 'tasks' given on input may be either a list of independent tasks or a dependency graph. 'transport' is any rsh/ssh-like program. Cool features: resistance to network failures, minimization of total calculation time.

В последнее время все большее распространение получают компьютерные системы оснащенные большим количеством процессоров или ядер. В настоящее время многоядерными процессорами комплектуются не только мощные сервера и рабочие станции, но также ноутбуки, нетбуки и даже мобильные телефоны и планшеты. В связи с этим часто возникает задача распараллеливания выполнения задач с использованием всех имеющихся вычислительных ресурсов. Та же проблема возникает при использовании кластера компьютеров, объединенных в вычислительную сеть. Задача не нова, и для ее решения имеется масса инструментов, таких, например, как MPI, стандартный API (реализован в библиотеках openmpi, mpich и др.), широко применяемый математиками для расчетов на супер-ЭВМ и кластерах. Тем не менее, существующие инструменты не лишены недостатков. В силу лицензионных ограничений далеко не всегда имеющиеся инструменты можно легально использовать в коммерческих целях, часто они имеют весьма специфическую область применения и неудобны для решения простых пользовательских задач, существующие инструменты порой слишком требовательны к количеству оперативной памяти и дискового пространства, а то и просто ограничены конкретной программно-аппаратной архитектурой.

Задача, ставшая в свое время перед автором — обработка больших массивов информации, а позднее обработка дерева задач, с ис-

*Минск, Беларусь, cheusov@netbsd.org

пользованием нескольких компьютеров различной аппаратной архитектуры. При этом «задача» в моем случае решалась автономной программой, написанной с применением различных языков программирования. Не найдя готового решения, подходящего для моего случая, я разработал программный пакет paexec с открытым исходным кодом (лицензия MIT), и разместил его в сети Интернет для публичного доступа.

Домашняя страница проекта: `http://paexec.sf.net/`.

Как это работает

В программный пакет paexec входят на данный момент две программы, главная из которых – paexec(1), собственно инструмент для распараллеливания, получающий в качестве аргументов

1. *задачи* для выполнения. Каждая отдельная задача — это произвольная текстовая строка, не содержащая пробелов. Это может быть, например, имя файла, формула для вычисления и т.п. Совокупность же задач может представлять собой множество независимых задач, то есть задач, которые могут выполняться параллельно, либо орграф, узлами в котором являются задачи, а ребро (A, B) означает: «для выполнения задачи B необходимо сначала выполнить задачу A; если задача A по каким-либо причинам не смогла быть выполнена, пометить задачу B и все другие задачи, зависящие от A как невыполненные». В целях задания порядка выполнения задач, каждой их них можно поставить в соответствии некоторый вес, определяющий, в какой момент лучше начать ее обработку. Реализовано несколько механизмов учета данных весов. В качестве веса можно, например, выбрать приблизительное время выполнения задачи или ее важность.
2. *список узлов*, на которых, будут производится вычисления, узлами могут быть, например, имена компьютеров в сети или номера процессоров, адреса chroot пространств на UNIX-системе и т.д.
3. *вычислитель*. Это программа, принимающая на вход задачу для вычисления и печатающая на стандартный поток вывода результаты в определенном формате.
4. *транспорт*, задающий способ связи с узлами, им могут быть такие программы как rsh, ssh или любые другие с совмести-

мым способом использования. В случае, если транспорт не указан, *вычислители* будут запущены локально указанное число раз.

На стандартный поток вывода `paexec(1)` выдает результат обработки задач вычислителями (последовательность текстовых строк) в порядке их поступления от вычислительных узлов (sliced output). Вторая программа, входящая в комплект — `paexec_reorder(1)`, она предназначена для пересортировки выходного потока `paexec(1)`.

Важно отметить, что `paexec(1)` устойчив к сбоям в сети, а это значит, может использоваться даже в сетях с неустойчивой связью, таких, например, как Интернет. Устойчивость заключается в том, что в случае потери связи с узлом, переданная ему на выполнение задача перераспределится на другой свободный узел в момент его появления. Периодически происходит попытка восстановления связи с узлами, связь с которыми была потеряна.

Success stories

На базе программного пакета paexec разработана распределенная система сборки программных пакетов pkgsrc (pkgtools/distbb), успешно применяемая в течение многих лет на таких операционных системах как NetBSD, Linux, Solaris, FreeBSD и др. Также, paexec успешно применяется в компании Invention Machine для распределенной обработки данных.

Кроссплатформенная подготовка и генерация отчетов средствами Qt и OpenOffice.org

Роман Гаранин*

The report is devoted to ways of report generation by means of Qt and OpenOffice. Existing variants of generation are considered. Indispensable conditions: open sorce, crossplatform, convenience of preparation and generation to the developer and to the user, absence of lacks of existing solutions.

Подготовка результирующих отчетов является по сути результатом подавляющего большинства бизнес-процессов на предприятиях. Цель данного доклада — показать некоторые способы упрощения подготовки и последующей генерации отчетности с минимальными временными и денежными затратами. Используемый при этом инструментарий является кроссплатформенным и полностью открытым, что дает заказчикам и разработчикам полную свободу действий.

Кроме того, такая система подготовки отчетов должна быть максимально простой и понятной для офисного сотрудника и не должна вызывать остановки работы комплекса автоматизации предприятия. Последний недостаток характерен для популярного семейства 1С:Предприятие, где обновление отчетности требует завершения работы в системе всех пользователей, что очень неудобно на крупных и средних предприятиях.

Подготовка макетов

Начальным этапом при подготовке непосредственно отчетов является создание макетов. По сути это разметка макета — определение статических и динамических областей. Статические области вносятся сразу же в макете и в дальнейшем не модифицируются. Динамические области зависят от типа и объема выходных данных. Для этих целей будем использовать табличный редактор:

*Брест, garanin.r@gmail.com

OpenOffice.org Calc. Calc имеет целый ряд преимуществ по сравнению с другими редакторами: кроссплатформенность, открытость, работа с открытым текстовым форматом, основанным на XML — OpenDocument, расширяемость, простота освоения и др.

Для определения ячеек, в которые необходим вывод данных можно воспользоваться идеей, взятой из 1С:Предприятие. Ячейки могут представлять собой:

- выражения — вывод непосредственно переменных;
- шаблоны — вывод смешаных статических и динамических данных заданных строкой при использовании специального синтаксиса.

Вывод возможен в ячейки по адресам, однако такой способ неудобен тем, что при вставке нескольких строк нижние ячейки сдвигаются вниз и их адрес меняется.

В OpenOffice.org поддерживается именование ячеек. Т. е. задав имена ячейкам ирограммно в них можно выводить данные.

«Движок»

В качестве «движка» обработки, подготовки и вывода данных в макет будем использовать программы на C++ с использованием библиотеки Qt от Nokia. Предпочтение C++/Qt в данном случае отдано потому, что на разных платформах скомпилированная программа выполняется гораздо быстрее, занимает меньше места, не требует установки огромного количества различных библиотек (достаточно нескольких файлов вместе с исполняемым). С выходом версии 4.5 возможности библиотеки значительно расширились. Тем не менее, генерация отчетности при разработке бизнес-приложений на Qt остается одним из краеугольных камней. В принципе, Qt несет «на борту» весь необходимый инструментарий при подготовке печатных форм отчетности, если речь идет о несложных отчетах и небольших программных продуктах, написанных на Qt. Для несложных отчетов может также использоваться популярная разработка NCReport.

Использование C++/Qt также позволит обеспечить удобную выборку данных из различных источников: из базы данных, GUI-форм при вводе данных пользователем, XML-данных, неструктурированных текстовых данных и др.

Варианты взаимодействия

Существует несколько вариантов взаимодействия программы на C++/Qt с шаблоном отчета OpenOffice.org Calc:

1. платформозависимые: взаимодействие через COM/OLE/ActiveX — распространенный вариант, но в данном случае нам он не интересен;
2. платформонезависимые:
   (a) взаимодействие через UNO;
   (b) правка XML-структуры файла OTS/ODS инструментарием C++/Qt;
   (c) управление генерацией через формирование макроса для OpenOffice.org

Взаимодействие через UNO (Universal Network Object)

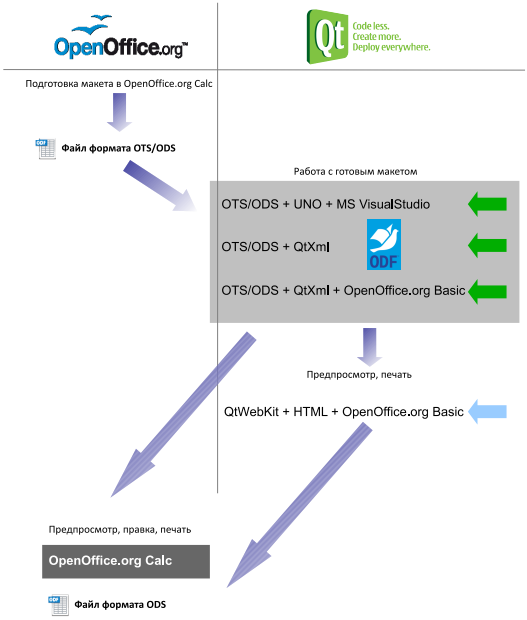
При использовании данного метода взаимодействия необходимо наличие OpenOffice SDK для сборки программ. Недостатком данного метода является ориентация OpenOffice SDK на работу с компиляторами от Microsoft для MS Windows. При использовании готовых собранных библиотек разработчиком это не является недостатком, тем более что Qt возможно интегрировать в Visual Studio. Сборка возможна также с помощью бесплатного набора Microsoft VC Toolkit 2003. Фактически, разработка такой программы, если планируется ее использование на различных платформах существенно затруднена.

Правка XML структуры файла OTS/ODS инструментарием C++/Qt

Богатые возможности Qt для работы с документами XML позволяют непосредственно править основные файлы документа OpenOffice.org, которые по сути представляют собой несколько XML-файлов, запакованных в zip. Этот способ не требует никаких дополнительных SDK, однако требует существенных трудозатрат при разработке: готовых библиотек на C++ для работы с файлами формата OTS/ODS пока очень мало и большинство из них находятся в стадии разработки (для Java такой иструментарий уже есть — это ODF Toolkit).

Управление генерацией через формирование макроса для OpenOffice.org

Этот способ на данный момент является одним из самых удобных. Суть его в том, что посредством инструментария C++/Qt для работы с XML в файл OTS/ODS добавляется сгенерированный программно текст макроса на поддерживаемом OpenOffice.org языке программирования. Мы будем использовать OpenOffice.org Basic, хотя возможны Python и JavaScript. Также в макрос добавляется метод, который срабатывает при открытии документа и запускает выполнение записанного из программы на C++/Qt макроса. Т. е. всю работу выполнит сам OpenOffice при загрузке документа. Недостатком данного способа является то, что у пользователя могут быть запрещены макросы.



Базовая серверная архитектура для высоконагруженного стартапа

Михаил Пянко*

This article describes how to create a good environment for start-up project: what kind of server do we need for each role, and how these servers should communicate. Architecture restrictions are considered. Technical and architecture suggestions are made.

На начальном этапе разработки стартапа многие разработчики сталкиваются с проблемой создания достаточно гибкой и в будущем масштабируемой архитектуры приложения, оптимизированной для большой нагрузки. Мы хотели бы поделиться опытом в этой области. В качестве примера рассмотрим методы построения ресурса, базирующегося на Ruby On Rails. Тем не менее примеры, которые будут приведены ниже, с легкостью могут быть использованы для LAMP-решений.

Серверы можно разделить на несколько типов по ролям:

- Application
- Database
- Load Balancer
- Utils
- Tools

Application — это frontend-сервер. Возможно использование как одного, так и нескольких Application-серверов. В случае использования одного Application-сервера надобности в балансировке нагрузки нет.

Application сервер выполняет следующие функции:

- обработка запросов от пользователей (получает запросы от балансировщика нагрузки),
- взаимодействие с базой данных (Database server),
- отправка сообщений в очередь сообщений (Utils server),

*Минск, ООО Анакреон, mihail.pianko@warecorp.com

– взаимодействие с CDN,
– взаимодействие с search engines (Util server),
– инвалидация кэша,
– работа с кэшем (Database server).

Application сервер не должен производить следующих действий:
– преобразование медиа контента (видео, аудио, слайдшоу, и т.д.),
– преобразование/кропинг картинок (если позволяет архитектура),
– отправка сообщений электронной почты (если позволяет архитектура),
– запросы на сторонние ресурсы через RestClient, curl, etc. (если позволяет архитектура).

Database — сервер, на котором находится база данных. В некоторых случаях на Database-сервер также устанавливается memcached, но это зависит от загрузки и конфигурации Database-сервера.

Database-сервер выполняет следующие функции:
– обработка запросов от Application и Utils серверов,
– хранение кэша (в случае установки memcached на DB сервер).

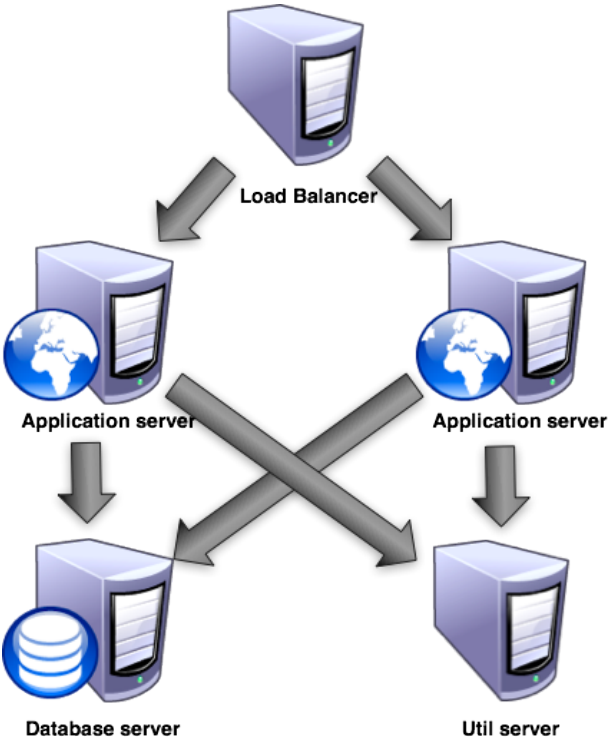
Load Balancer — балансировщик нагрузки, необходимый для распределения запросов пользователей между Application-серверами. Простейшим решением является использование nginx в качестве балансировщика нагрузки.

Load Balancer выполняет следующие функции:
– распределение запросов между Application серверами,
– обработка HTTPS соединений (SSL сертификат должен быть сконфигурирован на балансировщике),
– кэширование при использовании Varnish или связки nginx + memcache,
– Firewall.

Utils — сервер, на котором располагаются сопутствующие сервисы, необходимые для отложенной обработки данных. Вот краткий список:
– message queue (RabbitMQ, Apache Message Queue, etc),
– search engines (Solr, Sphinx, etc),
– обработчики сообщений,
– сервисы для обработки Video, Audio, SlideShow,
– отправка сообщений электронной почты,
– взаимодействие с CDN,
– инвалидация кэша,

– взаимодействие со сторонними ресурсами (PDF converters, SendGrid, etc.).

Tools — сервер, необходимый для установки сторонних решений для работы приложения. В частности, приложения, базирующиеся на Java/Tomcat, система мониторинга и ресурсы, в безопасности которых вы сомневаетесь (Spellchecker, etc.). Tools-сервер не может установить соединения с Application, Database, Utils-серверами. Единственный протокол взаимодействия — это HTTP. Это дает гарантию того, что в результате взлома стороннего компонента основная ферма не пострадает. Сервер с этой ролью не обязателен.



Ограничения, которые накладываются на архитектуру приложения

Хранение кэша

Необходимо организовать централизованное хранилище кэша. Файловый кэш и кэширование данных в памяти сервера работать не будет, так как кэш может создаваться и инвалидироваться с разных серверов. Наилучшим решением в данном случае является использования memcached.

Хранение сессий пользователя

В данном случае целесообразно хранить сессии в memcached или в Cookies. Отдача сессии на сторону пользователя — менее безопасное и более медленное решение, чем хранение в memcached.

Хранение медиа-данных пользователя

Любой контент, загруженный пользователем на сервер, не может быть просто сохранен в папку /public/upload. Наилучшим способом хранения данных является CDN.

Рекомендации по автоматизированной конфигурации серверов

Существует достаточно большое количество ПО для поддержания конфигурации серверов: Cfengine, Puppet, Chef, Sprinkle.

Для автоматизированного конфигурирования серверов мы используем Sprinkle. Sprinkle — это достаточно простое приложение на RoR, позволяющее разрабатывать сценарии по установке ПО на UNIX-сервера. Sprinkle поддерживает разделение серверов на роли, фермы и т.д. Так же поддерживаются различные типы инсталяторов, методов проверки текущей конфигурации, источников данных. Sprinkle имеет открытый исходный код, что позволяет вносить изменения в логику работы отдельных компонентов или создавать дополнительные модули.

Варианты инсталляции

Существует несколько вариантов ферм.

- Single server installation.
  - На одном сервере располагаются Application + Util + Database
- Light farm
  - 1 сервер: Application
  - 1 сервер: Util + DB
- Big farm:
  - 1 сервер: Load Ballancer
  - x серверов: Application
  - $1 - x$ серверов: Util
  - $1 - x$ серверов: Database (о репликациях и масштабировании базы данных можем поговорить чуть позже)

На самом деле существует множество комбинаций распределения ролей по физическим серверам. Необходимо выбирать оптимальный вариант в зависимости от нагрузки на сервера каждой роли.

Рекомендации по архитектуре. Несколько примеров

Интеграция со сторонними ресурсами

Зачастую необходима интеграция со сторонними ресурсами для преобразования видео и аудио, обработки слайд-шоу, отсылки писем и т.д.

Это подразумевает установку соединения по протоколам TCP, HTTP, etc. В данном случае существует несколько рисков:

- сервер недоступен
- сервер перегружен

В первом случае пользователь не сможет завершить действие, часть данных может быть потеряна или пользователь увидит сообщение ошибки на экране. Во втором случае мы получим очень долгую обработку страницы, что доставит неудобство пользователю.

Лучшим решением в данном случае является отправка события в очередь сообщений и обработка на стороне util-сервера.

Обработка картинок пользователя

Если ваш ресурс поддерживает возможность загрузки аватаров пользователей, логотипов с последующим ресайзингом изображений, вы можете столкнуться с проблемой большой нагрузки на сервер.

Есть несколько распространенных методов ресайзинга изображений:

- оригинальное изображение сохраняется в хранилище без обработки. Доступ к изображению со страниц сайта реализован через врапер с элементарной логикой. Если картинка сгенерирована, то необходимо отдать ее через STDOUT или вернуть редирект на статический. Если картинка отсутствует, то сгенерировать и выполнить предыдущий шаг. Проблема этого метода в том, что нагрузка, которую может создать данный алгоритм, не может быть спрогнозирована. При добавлении нового размера изображения достаточно визита поискового бота - и скорее всего сервера будут перегружены.
- оригинальное изображение преобразуется в нужные форматы и размеры непосредственно после загрузки на сервер. Это дает единовременную нагрузку на Application-сервера. Минусом данного метода является задержка в возврате страницы пользователю, что может вызвать дискомфорт.
- последний вариант является модифицированной версией предыдущего. Вместо непосредственной обработки изображения на Application-сервере необходимо отправить событие в очередь сообщения и произвести обработку изображения на Util-сервере.

Заключение

Выше приведены приемы для построения начальной архитектуры высоконагруженной системы, которая в будущем может быть легко расширена и смасштабирована. Способы дальнейшего развития напрямую зависят от архитектуры приложения, нагрузки на отдельные узлы системы и специфики проекта.

Мониторинг Linux/FreeBSD серверов

Николай Маржан*

Purposes and functions of monitoring systems are discussed, as far as general approaches to analyze the diagnostic parameters of a server. Observed information is classified into parameters received from hardware maintenance, operating system, services and typical database. Sample diagnostic parameters are presented for each category.

К стандартным целям проводимого на серверах мониторинга можно отнести:

- Отслеживание критических значений диагностических параметров состояния и уведомление инженеров об их появлении.
- Накопление статистической информации для последующего анализа.

Уведомление о том, что система находится в критическом состоянии, является наиболее важной функцией мониторинга, так как требует немедленного вмешательства инженеров, которые занимаются поддержкой данной системы.

Даже если проблема носит временный характер (например, стопроцентная загрузка процессора, возникшая спонтанно и затем прекратившаяся через 20 минут), выяснить причину такого поведения можно только тогда, когда проблема присутствует (причиной могла быть удаленная DOS-атака).

Накопление статистической информации происходит обычно в 2-х видах:

- Лог-файлы. Обычно в них записываются только те моменты, когда система входит в критическое состояние, и когда возвращается в нормальное.
- Данные для графиков по каждому показателю.

Графики — не менее важная часть системы мониторинга, так как только они позволяют проводить анализ состояния сервера в целом, как сложной системы. Используя графики для наблюдения динамики диагностических параметров, мы можем:

*Киев, Украина, PortaOne, Inc. delgod@delgod.com

– Увидеть, в каком состоянии находился сервер в любой момент времени в прошлом. Мы можем быстро понять, чем отличается сегодняшнее состояние от состояния недельной давности.
– Проследить закономерности возможной взаимосвязи между диагностическими параметрами. Например, когда повышается входящий трафик на сетевом интерфейсе — повышается нагрузка на центральный процессор.
– Находить противоречия между значениями диагностических параметров. Например, сильно возросло количество запросов на чтение к базе данных, и база данных начала использовать на 1 Гб больше оперативной памяти, но при этом с жесткого диска было считано всего 100 Мб данных. Увидеть, в каком состоянии находился сервер в любой момент времени в прошлом. Мы можем быстро понять, чем отличается сегодняшнее состояние от состояния недельной давности.
– Проследить закономерности возможной взаимосвязи между диагностическими параметрами. Например, когда повышается входящий трафик на сетевом интерфейсе — повышается нагрузка на центральный процессор.
– Находить противоречия между значениями диагностических параметров. Например, сильно возросло количество запросов на чтение к базе данных, и база данных начала использовать на 1 Гб больше оперативной памяти, но при этом с жесткого диска было считано всего 100 Мб данных.

Диагностическими параметрами состояния для аппаратного обеспечения являются:

1. температура центрального процессора;
2. наличие повреждённых секторов или ошибок чтения на жестком диске;
3. состояние аппаратного RAID-контроллера.

К диагностическим параметрам состояния для операционной системы можно отнести:

1. количество задач, ожидающих выполнения (load average);
2. время простоя центрального процессора (idle);
3. время простоя каждого ядра центрального процессора (idle);
4. размер свободного места на жестком диске;
5. нагрузка на жесткий диск;
6. количество данных в разделе подкачки (swap);
7. активная запись или чтение с раздела подкачки (swaping/paging);
8. количество использованной памяти;

9. количество памяти, потребляемой каждым приложением;
10. скорость передачи данных на сетевых интерфейсах;
11. подтверждение того, что сетевые интерфейсы работают на нужной скорости и в полнодуплексном режиме;
12. количество ошибок и коллизий на сетевых интерфейсах;
13. состояние системных буферов (freebsd vm.zone), mbuf clusters, sendfile-буферов, размер KVM, Pipe KVA;
14. количество открытых файлов, количество открытых сокетов;
15. отсутствие процессов-зомби;
16. отсутствие ошибок сегментации у процессов.

Диагностическими параметрами состояния сервисов являются:

1. подтверждение того, что все критически важные сервисы запущены.
2. подтверждение того, что сервисы успевают вычитывать данные из входящей очереди сетевой подсистемы.
3. подтверждение того, что нет отброшенных данных вследствие полных буферов соединения (dropped due to full socket buffers).

Диагностическими параметрами состояния базы данных MySQL являются:

1. количество максимально использованных соединений;
2. количество запросов, количество медленных запросов;
3. количество операций записи/чтения с жесткого диска;
4. отставание репликации.

Как показал опыт построения систем мониторинга в гетерогенных сетях, подходы и диагностические параметры практически идентичны для операционных систем FreeBSD и Linux; основные различия связаны с мониторингом системных буферов и с некоторыми расхождениями в синтаксисе команд.

Darktable, приложение для каталогизации и обработки RAW-файлов

Константин Шевцов*, Александр Рабцевич†

Darktable is an open source application devoted to processing of RAW files. The program can manage collections of RAWs with rating, color labels and custom tags. A rich set of built-in filters (some of them are unique), used at processing, store their settings in a form of a history stack, which is saved alongside original RAW, providing original RAW untouched. Thanks to rawspeed RAW importing library, multi-threading and permanent optimizations, the program is fast and responsive.

История

До появления darktable в ОС Linux отсутствовал открытый инструмент, сочетающий возможности каталогизации коллекции RAW файлов с их неразрушающей обработкой. UFRaw и Rawtherapee не обладают достаточными возможностями для профессионального фотографа.

Восполнить пробел решил Йоханес Ханика (Johannes Hanika), который зарегистрировал проект darktable на SourceForge в феврале 2009 года. Вскоре к нему присоединились 3 разработчика: Хенрик Андерсон (Henrik Andersson), Паскаль де Брайн (Pascal de Bruijn), Александр Прокудин. В настоящее время у проекта около 20 контрибьюторов.

Описание

Список основных возможностей программы при работе с коллекцией включает:

– импорт фотографий в коллекцию из папки, импорт отдельных фотографий. Фотографии физически не перемещаются.

*Новополоцк/Минск, Беларусь, kanstantsin.sha@gmail.com
†Минск, Беларусь, alexander.v.rabtchevich@gmx.net

– импорт из фотоаппарата посредством gPhoto2
– хранение данных о коллекции в собственной базе данных присвоение фотографиям рейтинга (stars), система цветовых меток, пользовательские теги (метки).
– поиск по произвольной комбинации: съемка, камера, метка, дата, наличие изменений после экспорта.
– копирование истории обработки между фотографиями
– экспорт в jpeg, tiff (8 и 16 бит)
– экспорт в Picasa, Flickr, email

Кратко перечислим основные возможности программы по обработке RAW (в версии из репозитория git). Программа построена на основе модульной архитектуры (т.е. плагинов). Внутреннее представление данных может быть RGB float (32 бит/канал) или LCh, в зависимости от модуля. Поддерживаются многопоточность и использование OpenCL для ряда операций, при наличии драйвера и подходящего графического ускорителя. В darktable реализован быстрый предосмотр с масштабированием вплоть до 100%, при котором обрабатывается только часть изображения, показываемая в окне с кешированием операций. Все изменения хранятся в виде стека истории с возможностью отката до произвольной точки, а также копирования истории изменений между фотографиями/или сохранения ее в виде стиля для последующего применения. Стек истории сохраняется в виде файла *.xmp вместе с оригинальным файлом RAW. Все модули поддерживают возможность создания и ручного/автоматического использования предустановок. Логически модули можно разделить на следующие категории: основная, цвет, корреция и эффекты. К модулям работы с цветом относятся:

– вельвия;
– регулирование яркости/насыщенности или сдвига тонов в плагине цветовые зоны;
– микшер каналов;
– алгоритмы дебайеризации ppg и AMaZE;
– поканальный баланс белого через множители либо через сдвиг цветовой температуры и множителя зеленого канала;
– редактируемая тональная кривая камеры с возможностью выбора из набора готовых кривых, аналогичных используемым производителями цифровых фотоаппаратов;
– восстановление пересветов при отрицательной экспокоррекции;

– профили камеры: стандартные (Adobe), улучшенные, либо пользовательские.

Модули коррекции изображения включают:

– трансформации: кадрирование с выбираемым соотношением сторон, поворот, перспектива;
– исправление геометрических искажений оптики с помощью библиотеки lensfun;
– редактируемая тональная кривая (L канал в LCh пространстве;
– возможность работать с тональной кривой в виде последовательности зон Адамса;
– мощнейший инструмент — эквалайзер, позволяющий регулировать локальный контраст (L и C) в зависимости от пространственной частоты (аналога радиуса в USM) с помощью избегающих краев вейвлетов;
– подавление шума.

И, наконец, основные модули эффектов:

– обесцвечивание с регулируемым цветовым фильтром;
– градиентный фильтр (GND) с возможностью поворота и радиального смещения;
– эффект виньетирования с регулируемыми на холсте радиусом, формой (окружность/эллипс);
– смягчающий фильтр.

Будущее

В настоящее время разработчики готовят программу к выпуску версии 0.9. В ближайших планах разработчиков — добавление функционала масок (Henrik Andersson) и переработка интерфейса с централизованной обработкой горячих клавиш (GSoC студент Robert Bieber). В состоянии обсуждения — возможность не однократного использования плагинов. В более отдаленных планах — использование GEGL внутри darktable с возможностью передачи данных в GIMP и обратно для более сложного редактирования; этот функционал может появиться не раньше выхода GIMP 3.0.

Проект активно развивается и радует постоянно растущее сообщество пользователей новыми выпусками с периодичностью примерно раз в три месяца. Можно уверенно заявить, что текущая версия пригодна для профессиональных фотографов.

Список литературы

[1] Официальный сайт проекта — `http://darktable.sf.net`

On Digital Monies

Daniel Nagy*

Payment over digital communications networks has been a hot topic since the introduction of the telegraph, the first such electronic network; originally, Western Union was a telegraph company. Money, however, is more than a means of payment: it is also an instrument of saving and credit, a measure of value and a unit of accounting. In my presentation, I shall explore various attempts at creating digital money, both proprietary and open-source, theoretical and practical.

The main difference between digitally represented information and everything else that has been historically used as money is that unlike the latter, digital information can be copied at no cost, with perfect fidelity in essentially unlimited quantities. Very undesirable properties for something that we intend to use as money. This is the main technical challenge in using digital information as money (known as 'double spending'). How can we make sure that no more gets spent than what has been earned?

There is, however, a different, no less interesting challenge that is not purely technical: how to get actual people provide real value in exchange for pieces of digital information. Why would people use pieces of digital information as a means of exchange or to keep their savings or to measure value?

The third challenge is a legal one. Since the first ever gold coins have been minted at the behest of Croesus, King of Lydia in the sixth century, B.C., money has been subject of government regulation and very often exclusive monopoly. How can digital money exist within the current legal framework?

Different solutions to these challenges have been proposed. We shall explore the solutions to these challenges in the following systems in some technical detail:

DigiCash

David Chaum's company founded in 1990 on the basis of some clever cryptography — blind digital signatures — he published in 1988. The

*Budapest, Hungary, ePoint Systems Ltd., nagydani@epointsystem.org

company declared bankruptcy in 1998, but in many ways it provided inspiration to the many ventures that followed. I brief and simplified introduction to the technology will be given as well as what I think were the reasons for failure.

WebMoney

Artiom Genkin went the other route: his doctoral thesis on the topic followed business success. Founded in 1998, WebMoney made its name by proving to be more resilient in the face of financial meltdown than the Russian banking sector. To this day, WebMoney is one of the most sophisticated and successful digital money, albeit mostly in the Russian-speaking world.

BitCoin

This is an open-source project started in 2009 by the mysterious Satoshi Nakatomo, based on earlier research either done or popularized by Nick Szabo (a former DigiCash employee). This year, BitCoin's popularity (and valuation) exploded, making it one of the most important and interesting innovations in the field.

ePoint

This is my own open-source project, kicked off in 2005. However, I am not going to talk much about how it works now; rather, I shall outline where I would like to take it and what needs to be done to get there. Maybe, like-minded developers of open-source software among LVEE 2011 participants will join me in creating something valuable and useful.

Linux и СПО в создании музыки и живом исполнении. Обзор имеющихся программ и ниши применения

Сергей Васильев*

The aspects of using Linux and free software in amateur music creation, sound editing and live performances are reviewed.

Ещё несколько лет назад Linux был малопригоден для работы со звуком; за последние годы ситуация значительно изменилась: улучшилась поддержка многих звуковых карт, MIDI-контроллеров и MIDI-интерфейсов, появились новые программы, интерфейс которых изменился в лучшую сторону, что снизило порог вхождения для начинающих пользоваталей. И, несмотря на некоторые всё ещё присутствующие проблемы, сейчас вполне возможно использование решений на базе Linux и СПО в любительской музыкальной деятельности. Применение данных решений имеет ряд преимуществ, одним из которых является низкая стоимость, что немаловажно для начинающих музыкантов.

Многие начинающие музыканты не обращают внимания на Linux из-за недостаточной информированности, и очень часто для выполнения простых задач прибегают к применению программного обеспечения, возможности которого на порядок превышают потребности, что приводит к нерациональному расходу денежных средств или к незаконному использованию коммерческих приложений. Имеющейся на данный момент программной базы и поддержки оборудования достаточно для выполнения основных задач.

В таблице приведены категории и некоторые программы, которые, после сравнительного анализа, были отобраны как наиболее подходящие для использования в сетапе для демонстрации.

По субъективной оценке, наиболее заполнены категории многодорожечных аудиоредакторов, представленных мощнейшим проектом Ardour, и некоторыми более простыми; процессоров эффектов — Rakarrak и др.; также почти на 100% покрывает потребности замечательный редактор табулатур TuxGuitar.

*Таллинн, Эстония, zyxos2@gmail.com

Software category	Programs used in setup
Loopers	SooperLooper
Synthesizers	FluidSynth, amSynth, Yoshimi
Sequencers	Rosegarden, MuSE
MIDI filtering programs	midish
Drum machines	Hydrogen
Multi-track audio editors	Ardour, Jokosher
Effect processors	Rakarrak
Aggregators	Arpage

Драм-машины представлены практически одним проектом Hydrogen, но довольно мощным, лишь с небольшими пожеланиями усовершенствований в области мульти-лэйеринга.

Что касается луперов и арпеджиаторов — лидеры SooperLooper и Arpage предоставляют базовую функциональность, но хотелось бы надеяться на улучшения в будущем.

Категории с некоторым количеством недочетов — секвенсеры и MIDI-фильтры.

Проблемные категории — синтезаторы (несмотря на замечательные программы FluidSynth, amSynth и Yoshimi, ситуация с VST-плагинами все еще далека от идеальной) и автоаккомпаниаторы (поскольку MMA очень сильно нехватает адекватного GUI).

Amazon auto-scaling. Создание и оптимизация автомасштабируемых систем на базе Linux

Виктор Краев*

The technology of auto-scaling clasters provided by Amazon for its cloud services is reviewed. Approach to monitor and control such cloud-based system is presented, as far as practical experience of migrating to the auto-scaled services.

Amazon EC2. Cloud vs Hardware

С каждым годом IT-индустрия набирает обороты, но как бы не развивались технологии, ресурс выработки железа пока никто не победил. Отказ оборудования по-прежнему является одной из главных проблем современного хостинга. Конечно же, всегда есть возможность создания избыточности, но это тянет за собой достаточно серьезные расходы на дополнительное оборудование и его обслуживание. Поэтому наша компания обратила внимание на облачные вычисления. Наиболее продвинутой компанией, способной решить наши задачи, на сегодняшний день оказалась Amazon. Хотя их технология Elastic Cloud — местами сырая, она находится в активной разработке и интегрируется с другими сервисами, нивелирующими ее недостатки. Не победив проблему «железа», в Amazon создали избыточность, покрыв недостатки другими сервисами, и сегодня падение инстансов (instances) или потеря данных в облаке иногда присутствует, но успешно компенсируется резервным копированием и высокой скоростью создания инстансов из резервных копий. Интеграция с такими сервисами, как auto-scaling, cloudwatch, cloudformation, позволяет добиться избыточности, намного превышающей возможности обычного кластера серверов, не задумываясь о «железе».

*Минск, Беларусь, anakreon.by

Amazon Auto-Scaling. Hardware cluster vs scalable cloud

На сегодняшний день для увеличения доступности и избыточности ресурса используют технологии кластеризации. Статический кластер подразумевает под собой использование определенного неизменяемого количества (больше 1) серверов, нагрузку между которыми распределяет балансировщик (Load Balancer). Это хорошее решение для ресурсов с постоянно высокой нагрузкой. Динамический кластер — это кластер, где распределение нагрузки осуществляется между серверами, количество которых может меняться в зависимости от нужд и нагруженности проекта. Однако в «железе» это трудноосуществимо, т.к. тянет за собой целый шлейф проблем, связанных с установкой оборудования, разворачиванием ПО, а главное — времени. С помощью сервиса EC2 на создание виртуального сервера уходит несколько минут. Создание происходит из уже готового образа (image), т. е. выполняется клонирование существующей системы. Обычно это образы «чистых» систем, в которых установлены лишь базовые пакеты. Но Amazon позволяет создавать собственные образы уже настроенных и готовых к использованию систем — например, хостинговых. Полученные образы помещаются в приватное хранилище, из которого в любой момент можно создать и отправить в работу клон. Эта технология, лежащая в основе создания динамического кластера и известная как AMI, позволяет мгновенно создавать кластеры различного размера. Для распределения нагрузки между элементами кластера используется сервис ELB (Elastic Load Balancer), который позволяет менять размерность кластера путем добавления или удаления виртуальных серверов в балансировщик. Однако, используя лишь web-интерфейс, вы сможете управлять кластером только вручную. Это значит, что если один из элементов кластера откажет в обслуживании, то балансировщик выбросит его из пула обрабатываемых запросов, и новый виртуальный сервер не встанет на его место автоматически. Чтобы автоматизировать этот процесс в Amazon придумали технологию Auto-Scaling. Эта технология позволяет задавать размерность кластера и использует проверку здоровья (health cheacking) серверов из ELB или EC2. В случае падения на место упавшего сервера из образа автоматически создается новый сервер и добавляет его в балансировщик. Сегодня управление конфигура-

цией Auto-Scaling осуществляется только через API, что позволяет автоматизировать процессы создания кластеров и управления ими, убирая из цепочки слабое звено — человека.

Amazon CloudWatch. Monitor and automate your systems

Мы можем не только управлять собственной армией виртуальных серверов, но и получать данные о ее работе. Технология CloudWatch позволяет осуществлять мониторинг. В основе этой технологии лежат метрики (metrics) — данные, собираемые с инстансов и агрегируемые по различным критериям в группы. Нельзя сказать, что технология блещет богатством собираемых коллекций, хотя в ней присутствует мониторинг CPU, дисковой и сетевой подсистем, количества запросов в ELB и т. д. Но с помощью API вы можете собирать свою собственную коллекцию метрик, которую в дальнейшем сможете использовать для автоматизации некоторых процессов. Автоматизация на основе метрик — главное предназначение этого сервиса. Кроме сбора статистики, этот сервис может ее обрабатывать и выполнять на основе полученных результатов определенные действия. В простейшем случае это отправка сообщения о превышении ранее выставленного порога. Кроме того, на определенное событие, полученное в ходе обработки метрик, можно прикрутить действия с Auto-Scaling, что дает возможность построения автоматически масштабируемыех кластеров.

Рассмотрим пример создания действия автомасштабирования с помощью API. Пусть требуется создать автомасштабируемый кластер — группу инстансов, обслуживаемых одним балансировщиком с минимальным количеством 1 и максимальным 7, которая меняет количество задействованных в распределении запросов на web-сервер в зависимости от нагрузки на процессор. Для этого мы создаем AMI-образ из инстанса, который мы хотим масштабировать. С помощью команды as-create-launch-config из AS API создаем конфигурацию, в которой указываем тип инстанса и ID нашего образа. После этого с помощью команды as-create-auto-scaling-group создаем группу, в параметрах которой указываем название нашего launch config, минимальный и максимальный размер группы, название ELB-балансировщика (который мы создали заранее). После этих действий кластер готов. Теперь необходимо интегрировать на-

шу AS-группу с сервисом CloudWatch. Для этого создаем 2 политики с помощью команды as-put-scaling-policy. Первая политика UP1 добавляет 1 инстанс в группу, вторая DOWN1 — удаляет 1 инстанс. Дальше, используя сервис CloudWatch (через web-интерфейс или API), создаем 2 обработчика событий:

1. При взлете выше среднего порогового значения 80% CPU за 5 минут — использовать политику UP1.
2. При падении ниже среднего порогового значения 20% CPU за 5 минут — использовать политику DOWN1.

Теперь кластер будет сам себя масштабировать, полагаясь на данные метрики CPU, агрегированной по нашей группе.

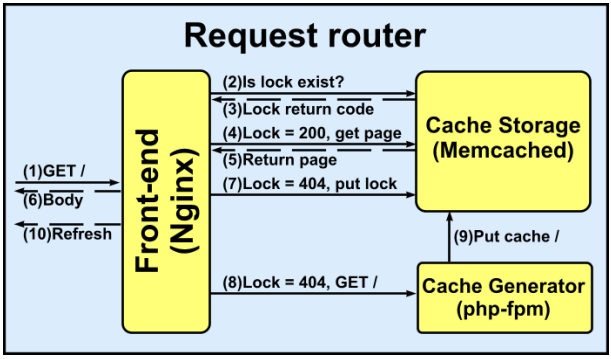
Nginx + Memcached over LUA. Cache and distribute

Технология автомасштабирования великолепно работает и решает повседневные задачи, обеспечивая бесперебойную работу интернет-ресурсов. Мы убедились в этом, осуществив миграцию интернет-ресурса `theuptake.org` в облако Amazon. Прежний кластер из 8 серверов стал автомасштабируемой системой. В обычном режиме на данный момент работает 1 small instance (самый дешевый виртуальный сервер), который в случае увеличения нагрузки масштабируется в течение нескольких минут до стабильного состояния между средними пороговыми значениями метрик CloudWatch. Однако, путь в динамическому масштабированию оказался не так прост, как использованиесервисов Amazon. Интернет-ресурс построен на движке Wordpress, который очень ресурсоемкий и требует значительного количества процессорного времени на обработку PHP-кода. Существует несколько кэширующих плагинов, решающих эту проблему, и мы опробовали их все. Из них можно выделить 2 мощных решения: WP Super Cache и WC3 Total Cache. Однако и для них результаты оказались недостаточными для динамического кластера. Самый быстрый — WP Super Cache. Он помещает сгенерированные страницы в файловую систему и (например, с помощью Nginx и реврайтов) позволяет отдавать контент с по-настоящему высокими скоростями. Однако его использование порождает 2 проблемы:

1. Каждый инстанс хранит свой уникальный кэш.

2. При добавлении нового инстанса в кластер кэш отсутствует, и требуется время на его генерацию, что в условиях быстро увеличивающейся нагрузки хоронит его под прессом запросов.

Плагин WC3 Total Cache, как и WP Super Cache, может хранить кэш в файловой системе, но умеет складывать страницы (и даже SQL-запросы) в memcached — общее для всех инстансов хранилище. Это решает проблему инвалидации и прегенерации кэша, но, скорости оказываются очень далеки от идеала. Если раньше ресурсы уходили на генерацию кода, то в данном случае они уходят на обслуживания системы кэширования, которая исполняется на том же самом PHP. Поэтому было принято решение сделать систему кэширования без использования PHP и связать Nginx напрямую с Memcached, исключив таким образом PHP из управления кэшем. Для реализации этой идеи потребовались Memcached и Nginx с модулями lua-nginx-module, memc-nginx-module, ngx_devel_kit, ngx_http_upstream_keepalive, set-misc-nginx-module, srcache-nginx-module. С помощью этого ПО автором была реализована приведенная на рис. схема.



По этой схеме Nginx выступает в качестве маршрутизатора между хранилищем кэша и кэш-генератором, вызывая для обработки запросов LUA-скрипт. Таким образом, страницы всегда отдаются из кэша. Если же кэша не существует, отправляется запрос для его генерации, а клиент перенаправляется на refresh-страницу, которая заставляет отправить запрос повторно через заданное время. Кэш имеет expire time = 0, что означает, что он никогда не будет удален. Для обновления кэша используется lock с ограниченным expire time (в зависимости от запрашиваемой страницы), и когда expire time ис-

текает — кэш регенерируется и помещается поверх старого. Также lock имеет еще одну важную функцию: его наличие предотвращает отправку запросов на генерацию из других фронтендов, если автомасштабируемый кластер стал раздуваться, и в группе находится больше одного инстанса.

Общее хранилище решает проблему прегенерации кэша. Так, при добавлении нового инстанса, он сразу вступает в распределение нагрузки, используя кэш из общего хранилища. Такой подход позволяет ускорить ресурсы не только с Wordpress, а в принципе любой ресурс с ПО, написанным на любом коде, из которого генерируется HTML.

Save and profit!

Использование данных технологий позволило нашей компании существенно снизить затраты на техническое обеспечение интернет-ресурса `theuptake.org`. Заменив 8 физических серверов (80–90% времени они простаивали, но были необходимы из-за периодических резких всплесков посещений после публикации новостей), на 1 large-инстанс, на котором хранится общий контент и кэш, 1 RDS-инстанс, в котором лежит БД и автомасштабируемая группировка, работающая во время простоя с 1 smallинстанса и раздуваемая до кластера, размер которого определяет нагрузка. В дополнение к экономии (т.к. сегодня мы платим только за пики) мы многократно повысили скорость отдачи контента, доступность и отказоустойчивость ресурса.

Способы повышения производительности высоконагруженных проектов на CMS Drupal

Виталий Иоскевич*

Drupal is a popular CMS/CMF used to power the different types of web-projects worldwide. The desire of developers to make their system competitive in terms of functionality and flexibility sooner or later leads to increase of server load and reduced performance. Extensive functionality, flexibility in configuration and extensibility through third-party modules are undoubtedly a huge advantage, enabling to carry out projects of varying complexity and purpose, but the price often is a poor performance of high loaded Drupal-powered sites. In our review we are going to revise existing standard methods of Drupal performance optimization (both server- and client-side) and demonstrate custom solutions that can be applied to some of the critical areas of Drupal-powered high-loaded website. In our real-life example we are going to show how to build Google Map page, capable to show thousands of markers based on Drupal content nodes.

Drupal (7-ая, текущая версия) является достаточно развитой системой управления контентом. Стремление разработчиков сделать свою систему универсальной и более функциональной, чем у конкурентов, рано или поздно приводит к тому, что возрастают нагрузки на сервер, снижается быстродействие. Обширный функционал, гибкость в конфигурировании и расширяемость посредством сторонних модулей несомненно являются огромным преимуществом, благодаря которому позволяет выполнять проекты различной сложности и назначения, но плата за него — низкая производительность готового решения.

Чем больше число использованных на сайте сторонних модулей (в т. ч. собственной разработки), тем ниже производительность сайта при больших нагрузках (большом количестве посетителей), а именно при построении сайтов с предполагаемым большим числом

*Минск, Беларусь

посетителей никак не обойтись стандартным функционалом CMS. Таким образом, данная CMS оказывается весьма требовательной к ресурсам сервера, и в большинстве случаев проблема решается выбором специализированных дорогих хостингов — выделенный сервер или несколько серверов.

Но и при ограниченных возможностях сервера есть способы улучшения производительности сайта. Стандартной системы кэширования Drupal достаточно для нормальной работы среднего сайта при неинтенсивной посещаемости, однако для случая высокой нагрузки на сайт этой системы недостаточно.

Основные методы повышения производительности Drupal:

- включить кэш в модулях, позволяющих кэширование (таких как Views, Panels, Feeds, SWF Tools и т.п.);
- увеличить время жизни кэша для стандартной системы кэширования Drupal. При этом надо иметь ввиду, что посетители будут видеть обновления содержимого значительно позже;
- если нет большой необходимости в статистике, можно выключить стандартные модули statistics и database logging. Они используют дополнительные запросы к БД при загрузке страницы. Кроме того, можно найти альтернативу данным модулям;
- использовать модуль CSS Gzip. Уменьшает размер файлов и количество http-запросов;
- использовать модуль Global Redirect. Предотвращает кэширование дубликатов содержимого сайта по ссылкам псевдонимам (синонимам) в случае использования «Clean URL»;
- использовать модуль Javascript Aggregator. Настройки Drupal позволяют объединить js-файлы, однако этот модуль позволяет еще и уменьшить конечный размер передаваемого файла;
- заменить системный cron на более функциональный модуль Elysia Cron. В отличие от системного, Elysia Cron позволяет распределить задачи по времени одна за одной, а не все вместе, при этом снижая пиковую нагрузку на сервер;
- перенести вызов javascript вниз станицы;
- помещать сторонние библиотеки вне каталога модулей /sites/all/modules — например, в /sites/all/libraries. В таком случае Drupal не будет просматривать ненужные ему файлы;

– конвертировать таблицы MySQL в UTF8 InnoDB;
– создать индексы для медленных запросов Views. Желательно проанализировать такие запросы и увеличить производительность созданием индексов для полей;
– использовать ImageMagick вместо стандартной GD image library, т. к. он лучше использует ресурсы сервера и дает лучшее качество изображений;
– использовать модуль Content Delivery Network integration. Он позволяет распределить файлы на множество разделенных серверов. При этом снижается динамическая нагрузка на сервер, но требуется дополнительная плата за каждый сервер;

Наиболее продуктивный способ увеличения производительности — отдельная эффективная система кэширования.

Boost. Одна из лучших систем кэширования для shared hosts. Создает копии html-станиц, которые генерирует Drupal, и хранит их в каталоге cache. Используя правила «.htaccess», Boost проверяет, существует ли необходимый файл. Если да, то загружает статичный html, полностью избегая Drupal/PHP/MySQL. Если нет, генерирует файл. Старые файлы удаляются по cron для того, чтобы выводимое содержимое сайта оставалось свежим (актуальным).

Memcache. Серверная система кэширования, сохраняющая страницы сайта в оперативной памяти сервера. Все что необходимо для эффективной работы в данном случае — быстрое ОЗУ большого объема. Его можно успешно использовать как для кэширования анонимных пользователей, так и для зарегистрированных. Но возможен и вариант, когда Memcache кэширует зарегистрированных пользователей, а Boost используется для кэширования анонимов. К преимуществам варианта можно отнести возможность кэширования анонимов и зарегистрированных пользователей, а к недостаткам — то, что нужен хотя бы VPS, с достаточной оперативной памятью и невозможность работы на shared hosting.

APC(Alternate PHP Cache). Система кэширования op-code для серверов Apache. Подход весьма эффективен при большом числе активных модулей на сайте. Суть системы заключается в кэшировании компилированного байт-кода PHP-скрипта для того, чтобы избежать расходов ресурсов сервера на разбор и компиляцию исходного кода при каждом запросе. Для дальнейшего повышения производительности кэшированный код хранится в общей памяти и непосредственно выполняется оттуда, тем самым сводя к минимуму количество медленных операций чтения файлов и копирования

в памяти во время выполнения. К преимуществам относится ускорение компиляции скриптов Drupal и уменьшение использования ресурсов сервера, к недостаткам — необходимость произвести специальное конфигурирование сервера.

Varnish (reverse proxy http-accelerator). Данная система выступает на первом плане над сервером Apache, PHP и Drupal. Varnish сохраняет кэшированное содержимое в оперативной памяти, и таким образом экономит ресурсы на загрузке Apache и Drupal. В результате этого Varnish предоставляет высокую производительность для кэшированных страниц для анонимных пользователей и является предпочтительным вариантом для сложных ресурсоемких сайтов, требующих значительной масштабируемости. Для работы с Drupal 6 необходима была отдельная «оптимизированная» сборка дистрибутива под названием PressFlow и установка модуля интеграции Varnish HTTP Accelerator Integration. Для текущей версии Drupal достаточно установки модуля.

Пример проекта с высокой нагрузкой — 350.org (Drupal 6)

К характеристикам проекта можно отнести следующие:
- 150 000+ уникальных посетителей в день
- 6000+ ивентов, созданных пользователями
- ок. 500 авторизованных пользователей постоянно на сайте
- самая посещаемая страница: карта ивентов (/map)

Кэширование критично как для анонимов, так и для авторизованных пользователей. Используется Pressflow/Varnish и специальные меры для повышения производительности:
- Отказ от Views для генерации больших массивов данных
- XML-файл на диск вместо DB-кэша (плюсы/минусы)
- Кэширование на диск всех RSSЭ=фидов
- Ajax-запросы информации для маркеров на карте.

Открытые протоколы — основа распределенных социальных сетей

Александр Загацкий*

Distributed networks help users to control their online identity in the Internet. Their key feature is absence of a central server. Nodes in the network belong to each one of its participants, and software runs on user's computer or on trusted server. User determines the distribution policy of his data. Distributed social networks are built upon open technologies, such as Atom, WebFinder, OpenID/OAuth, Salmon, ActivityStreams and etc. Most of developed distributed networks are opensourced and under development. Some of them are available to use, i. e. StatusNet — the open source microblogging platform that helps you share and connect in real-time within your own domain. Yet another distibuted social network OneSocialWeb is a dream of the world where all social networks are connected and work together similar to email.

Весной этого года в одном из дата-центров провайдера облачного хостинга Amazon EC2 произошел сбой [1], в результате чего перестали функционировать такие сервисы как Friendfeed, Quora, Reddit, Hootsuite, Foursquare. По неожиданной случайности это совпало с началом судного дня в сериале «Терминатор: Битва за будущее». По сюжету фильма в этот день военная компьютерная сеть SkyNet, управляющая военными роботами вышла из под контроля человека и произвела ряд ядерных атак на человечество. К каким последствиям может привести утрата контроля над персональными данными, созданными пользователями в сети и раскрывающие отдельные моменты их личной жизни?

Этот случай заставил задуматься о надежности социальных сервисов и сохранности пользовательских данных. Помимо этого за социальными сервисами стоят организации. Как следствие, функционирование сервиса зависит зачатую от материального положения владельца сервиса. Таким образом, сохранность личных данных и контроль над доступом к ним являются актуальной задачей для пользователей Интернет.

*Витебск, Беларусь, http://zag.ru, me@zag.ru

Отчасти решить очерченную задачу призваны децентрализованные (распределенные) социальные сети. Характерной их особенностью, является отсутствие центрального сервера. Узлы сети принадлежат каждому из ее участников, а программное обеспечение функционирует на компьютере пользователя или на доверенном сервере. При этом пользователь самостоятельно определяет политику распространения его данных.

В основе функционирования распределенных сетей лежат открытые протоколы. Ключевыми из них являются следующие:

Atom/RSS

Формат синдикации Atom основан на XML и позволяет описывать наборы веб-ресурсов. Например: новостные ленты, анонсы статей в блоге, галереи фотографий и тому подобное. Формат описан в RFC 4287 [2].

XRD

Стандарт описания ресурса доступен в [3]. Представляет собой последовательное описание данных аккаунта в формате XML. Среди данных — сведения о серверах аутентификации, публичные криптоключи и т.д.

WebFinder

Протокол, позволяющий идентифицироваться пользователю. В качестве идентификации используется запись вида

```
acct:joe@example.com
```

По данному адресу устанавливается сервер с учетной записью пользователя и извлекаются XRD-записи.

OpenID/OAuth

Протоколы аутентификации и авторизации. Позволяют повысить безопасность при взаимодействии нескольких независимых сервисов.

Pubsubhubbub

Протокол позволяет сообщать об обновлениях другим ресурсам. При этом используются так называемые хабы, которые сообщают

об изменениях на основном ресурсе подписчикам. Такая схема позволяет реализовать Push-канал обновлений, снизив нагрузку на источник изменений.

Авторами являются сотрудники Google: Brad Fitzpatrick and Brett Slatkin. Источник обновления децентрализуется. Адрес hub для обновлений указывается в RSS/Atom в следующем формате:

```
<link rel="hub" href="http://myhub.example.com/endpoint"/>
```

Salmon

Этот протокол позволяет отправлять действия по отношению к объекту (например, статье или заметке) с агрегаторов в оригинальный источник. Например, Salmon используется для сбора комментариев к оригинальной статье с других сайтов, на которых статья была опубликована.

Адрес для отправки обновлений указан в RSS/Atom:

```
<link rel="salmon"
href="http://example.org/salmon-endpoint"/>
```

Особенностью данного протокола является крипто-подпись сообщений, которая позволяет решить проблемы со спамом. Для получения публичного ключа используется формат XRD.

ActivityStreams

Формат описания активности пользователя на ресурсах. Благодаря этому формату происходит уведомление узлов распределенной сети о действиях пользователей: написании комментария, выставление оценки «нравится», публикации фотографий и т.д. Из форматов доступны как JSON, так и расширения Atom [4].

XMPP

Протокол обмена XMPP («Jabber») лежит в основе транспорта между узлами нескольких распределенных сетей. В качестве клиента используется как десктопные программы, так и мобильные версии.

Portable Contacts

Протокол безопасного предоставления данных о своих контактах внешним сервисам [5]. При этом используется протокол OAuth, и пользователь управляет доступом к своему списку контактов.

Среди проектов создания распределенных социальных сетей можно отметить следующие:

StatusNet [6]
: Открытую платформу для микроблоггинга (аналог Twitter) StatusNet можно установить на собственном домене. Платформа предназначена для коллективного использования. Лицензия: AGPLv3.

GNU Social [7]
: Соответствует философии Unix: маленькие программы делают маленькие работы. По возможностям близка к Status.net: микроблоггинг. Лицензия: AGPLv3.

Diaspora [8]
: Проект Diaspora, созданный по инициативе четырех студентов, предназначен для персонального использования: ведения микроблога, публикации фотографий. Лицензия: AGPL-3.0.

friendika [9]
: Возможности: сервер профилей и поддержка ряда других сетей (Diaspora, Google Buzz, Identi.ca, Twitter), поддержка шифрования между серверами. Лицензия: BSD License.

OneSocialWeb [10] Данная сеть микроблоггинга использует в качестве транспорта протокол XMPP [11]. В качестве идентификатора пользователя используется jabber ID, часть протоколов адаптирована для использования в Jabber-сети. Лицензия: Apache 2.

Это небольшая часть из разрабатываемых проектов, позволяющих пользователю сети управлять своими данными и быть независимым в контроле над персональной информацией. Тем не менее, очевидно, что персональные данные являются основной составляющей социальной жизни в сети и определять политику их использования может только пользователь.

Список литературы

[1] «Судный день для моих персональных данных». http://zag.ru/a4BR2

[2] RFC 4287: The Atom Syndication Format. http://tools.ietf.org/html/rfc4287

[3] Extensible Resource Descriptor (XRD). http://docs.oasis-open.org/xri/xrd/v1.0/xrd-1.0.html

[4] Формат оповещений Activity Streams. http://activitystrea.ms/

[5] Протокол Portable Contacts. http://code.google.com/intl/ru-RU/apis/contacts/docs/poco/1.0/developers_guide.html

[6] Платформа для микроблоггинга StatusNet. http://status.net/

[7] Распределенная сеть GNU Social. http://foocorp.org/projects/social/

[8] Социальная сеть Diaspora. http://www.joindiaspora.com/

[9] Распределенная сеть Friendika. http://project.friendika.com

[10] Социальная сеть OneSocialWeb, построенная на основе протокола XMPP. http://onesocialweb.org/

[11] XMPP — открытый протокол обмена сообщениями и информацией о статусе. http://ru.wikipedia.org/wiki/XMPP

Perl 6 Pod — формат ведения документации

Александр Загацкий*

Pod is an evolution of Perl 5's POD markup. Compared to POD, Perl 6's Pod is much more uniform, somewhat more compact, and considerably more expressive. Established in 2005, specification has undergone several revisions and is currently stable. The specification is written in Perl 6 Pod and is a good means of testing the implementations. There are several implementations in Perl 5 and Perl 6. Presentation covers the differences from Perl 5 POD, key features and experience of Perl 6 Pod personal usage.

Формат разметки Perl 6 Pod перестал быть черновиком. В финальной версии спецификации Synopsis 26 [1] добавился ряд особенностей. Никогда не еще не было настолько тесной интеграции программного кода и документации к нему.

Так, добавился специальный тип блоков документации — блоки-декларатоpы. Данные блоки предваряются специальным комментарием `#=`, за которым следует текст документации. Оформленные блоки документации ассоциируются с деларатоpами подпрограмм, а так же переменных. Уникальной особенностью является тот факт, что эти блоки документации становятся доступными через специальный атрибут `.WHY`. Например:

```
sub fu (           #= This text stored in &fu.WHY
 Any     $bar,    #= This text stored in $bar.WHY
 Mode    :$baz   #= This text stored in $baz.WHY
   ) { ... }

#= This is a special chainsaw
my SwissArmy $chainsaw    #= (It has a rocket launcher)

say $chainsaw.WHY; # prints: This is a special chainsaw
                   #         (It has a rocket launcher)
```

*Витебск, Беларусь, http://zag.ru, me@zag.ru

В приведенном примере текст документации к переменной `$chainsaw` становится доступным из программного кода. В указанном примере блок кода будет выведен на экран:

```
say $chainsaw.WHY; # prints: This is a special chainsaw
                   #         (It has a rocket launcher)
```

Уникальна сама концепция, а именно доступ программного кода к документации по нему. То есть становится возможным менять логику программы в зависимости от описанной в документации!

Еще одним примером интеграции документации и кода являются контекстуальные псевдонимы. Для их определения используется директива `=alias`. Например:

```
# This is actual code...
sub hash_function ($key)
=alias HASHCODE
{
    my $hash = 0;
    for $key.split("") -> $char {
        $hash = $hash*33 + $char.ord;
    }
    return $hash;
}
```

В данном примере определяется псевдоним HASHCODE. Его значением является следующий за ним программный блок. Теперь, чтобы привести в документации пример кода достаточно указать имя псевдонима внутри кода форматирования `A<>`:

```
=begin pod
An ancient (but fast) hashing algorithm is used:
=begin code :allow<A>
 A<HASHCODE>
=end code
=end pod
```

В результате в документации будет вставлен блок кода из программы:

```
An ancient (but fast) hashing algorithm is used:

    my $hash = 0;
```

```
for $key.split("") -> $char {
    $hash *= 33;
    $hash += $char.ord;
}
return $hash;
```

Это позволяет избежать ручного копирования блоков кода и последующую актуализацию документации.

Список литературы

[1] Спецификация формата Pod (Synopsis 26). https://github.com/zag/specs/raw/master/S26-documentation.pod

LeechCraft: открытый модульный интернет-клиент

Георгий Рудой*, Олег Линкин†

LeechCraft, the open source modular cross-platform internet client written in C++/Qt, is reviewed mostly from the technical perspective. Design decisions are described as far as solutions of the problems faced during development and arising from the modular nature of the application. Other aspects of open development are also briefly considered, such as building a community or future goals. The paper should be of high interest to those developing highly modular applications or general open source projects.

LeechCraft — это открытый модульный кросс-платформенный интернет-клиент. LeechCraft ориентирован на выполнение типичных пользовательских задач, связанных с интернетом, в том числе:

- просмотр веб-страниц;
- чтение лент новостей;
- скачивание файлов;
- обмен мгновенными сообщениями.

В то же время реализованы и другие функции: например, менеджер пакетов, работающий в пространстве пользователя, простой текстовый редактор, медиа-плеер.

Модульность приложения означает, что все задачи в LeechCraft выполняются загружаемыми модулями, которые при желании можно отключать либо не устанавливать. Модули также могут взаимодействовать друг с другом: например, модуль для чтения лент новостей использует модуль поддержки протокола HTTP для скачивания этих лент. Ядро приложения при этом обеспечивает лишь загрузку модулей, корректный порядок их инициализации, обеспечивает их взаимное взаимодействие и предоставляет некоторые базовые сервисы, например, сетевой кэш или базу данных cookies.

*Московский физико-технический институт, 0xd34df00d@gmail.com

†Министерство Обороны Республики Беларусь, MaledictusDeMagog@gmail.com

Среди преимуществ такой архитектуры со слабо зависящими друг от друга (а именно, от конкретной реализации) модулями можно отметить высокую гибкость приложения, значительное снижение дублирования функциональности и кодо-расширяемость, возможность заменять модули на другие, выполняющие аналогичные функции, без изменения кода вне этих модулей, и т. д. Так, например, модуль поддержки скриптовых плагинов для LeechCraft позволяет любому другому модулю взаимодействовать со скриптами, даже если этот модуль не имеет никакой прямой поддержки скриптинга.

Одним из выбранных способов реализации такой архитектуры является взаимодействие модулей при помощи обмена сообщениями — специальными структурами данных с заданными правилами обработки. Один модуль посылает сообщение по глобальной шине сообщений LeechCraft, а ряд других модулей опрашивается на предмет возможности обработки этого сообщения, и каждый из этих модулей может во время выполнения либо обработать сообщение, либо отказаться от его обработки.

В то же время иногда необходимо тесное связывание модулей друг с другом. Например, модуль блокировки рекламы в веб-браузере тесно связан с самой реализацией модуля-браузера, и сделать его абстрактным не представляется возможным. Аналогично, система поддержки стилей чатов в клиенте мгновенных сообщений связана с реализацией самого IM-клиента в LeechCraft. Для таких случаев обеспечивается возможность прямого соединения модулей друг с другом (при этом один из модулей считается плагином для другого модуля), а модули сами определяют удобный для них протокол взаимодействия.

Однако, помимо упомянутых преимуществ, поставленная задача реализации конечной функциональности при помощи слабо зависящих друг от друга модулей создает ряд проблем, часть из которых связана со статичной природой выбранного языка реализации (C++) и отсутствием в нем полноценной поддержки RTTI и какого-либо Reflection. Например, при реализации вышеупомянутого модуля поддержки скриптовых плагинов необходима возможность во время выполнения приложения определять используемые другими модулями протоколы взаимодействия друг с другом, так как прямое добавление информации о соответствующих протоколах на этапе написания кода либо на этапе компиляции противоречит принципу слабого связывания. Возможности интроспекции,

предоставляемые фреймворком Qt, позволяют успешно решить эту проблему при условии соблюдения простых правил при написании модулей.

Другие вопросы, возникающие при разработке подобных приложений — это создание сообщества вокруг приложения и его общее позиционирование. Социальный аспект открытой разработки является не менее важным, чем технический, поэтому он также обсуждается в докладе.

Список литературы

[1] LeechCraft Architecture — `http://leechcraft.org/development-leechcraft-architecture`

[2] Gamma et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995

Создание специализированного дистрибутива Linux для подключения к национальной GRID-сети

Денис Пынькин*

Unicore is ready-to-run system used as main platform for Belarusian national GRID network. There are no Linux distributions for bare-metal Unicore installation and user-friendly administration. Article describes steps to produce the ALT Linux based server with Unicore services working out of the box.

Сборка базового дистрибутива

Современные дистрибутивы ALT Linux собираются с помощью инструмента mkimage, который состоит из набора правил для утилиты make и вспомогательных скриптов и предоставляет базовый функционал для сборки образа Linux-системы. Фактически для создания своего собственного дистрибутива достаточно в правильном порядке и с правильными параметрами собрать из предоставляемых «кирпичиков» единое целое, что в общем-то является нетривиальной задачей.

Так было предпринято несколько попыток создать универсальные профили для mkimage, но наиболее удачным и развитым оказался mkimage-profiles-desktop (в рассылках и на форумах часто сокращают до аббревиатуры M-P-D).

Далее, на примере создания дистрибутива Unicore для упрощения подключения к белорусской национальной научной GRID-сети, описывается как можно использовать M-P-D для создания собственного инсталляционного образа сервера.

Для начала необходимо получить правила сборки дистрибутива из git-репозитория:

```
git clone git://git.altlinux.org/people/boyarsh/packages/
 mkimage-profiles-desktop
```

*Минск, Беларусь

Создать файл для пакетного менеджера apt с описанием репозитория и соответствующей архитектурой, который необходимо расположить по пути ~/$branch-$arch.conf, в случае стабильного репозитория для Platform 6 и 64-битной архитектуры — ~/p6-x86_64.conf. В качестве минимума в этом файле должна содержаться ссылка на файл с описаниями источников пакетов, для нашего примера — ~/p6-sources-x86_64.list.

Например для белорусского зеркала ALT Linux эти файлы будут выглядеть следующим образом:

p6-x86_64.conf:

```
Dir::Etc::SourceList "/home/d4s/p6-sources-x86\_64.list";
```

p6-sources-x86_64.list:

```
rpm [p6] ftp://ftp.mgts.by/pub/ALTLinux/p6/branch x86_64
 classic
rpm [p6] ftp://ftp.mgts.by/pub/ALTLinux/p6/branch noarch
 classic
```

Для создания установочного диска своего сервера удобнее всего использовать профиль от Антона Фарыгина, который можно собрать в качестве теста, чтобы проверить работоспособность конфигурации. Например, таким образом можно собрать установочный диск для 64-битной архитектуры с дизайном Sisyphus:

```
archs=x86_64 ./make-distro server-light
 --with-branding=altlinux-sisyphus
```

На консоль ничего не выводится, но это не страшно — «все ходы записаны», а ход сборки протоколируется в файлы вида <target>.<arch>.log или, в нашем случае, в файл server-light.x86_64.log.

Получившийся образ диска сохраняется в домашнем каталоге /out/<target> (/out/server-light/).

Теперь можно приступать к созданию собственного сервера.

Модификация базового дистрибутива

Модификация базового дистрибутива проводится с помощью дополнительных пакетов и изменения M-P-D для добавления своих

правил сборки дистрибутива и списков программного обеспечения, которое будет доступно во время и после установки.

Все дополнительные пакеты условно можно разбить на две категории — функциональные и вспомогательные. К функциональным относятся пакеты, которые содержат все необходимое для полноценного выполнения сервером той задачи, для которой он создается. Для создания управляющего сервера Unicore, технически готового для включения в белорусский национальный GRID, это пакеты, содержащие сами сервисы Unicore: gateway, unicorex, xuudb и TSI. Кроме того, для полноценного управления вычислительным кластером необходим сервис управления пакетными задачами Torque, а по требованию операционного центра добавлено программное обеспечение учета ресурсов и система мониторинга Ganglia.

Кроме того разработаны специально для дистрибутива два дополнительных пакета:

- unicore-grid-by — содержит конфигурационные скрипты и файлы, специфичные для белорусской GRID-сети.
- alterator-unicore-servers — минимальный интерфейс для настройки Unicore, реализованный в виде модуля к системе управления Alterator.

Вспомогательные пакеты необходимы только для корректной установки функционала, создания интерфейса либо на подготовительном этапе создания своего установочного диска.

Так можно выделить следующие пакеты:

- installer-distro-unicore-server — этот пакет содержит правила, которые будут выполняться во время установки на различных этапах. Например тут можно указать в каком порядке и какие шаги выполнять программе-установщику, какие сервисы будут запускаться. Сюда же можно добавлять скрипты для более тонкой настройки системы в процессе инсталляции.
- installer-feature-vm-unicore-server-stage2 — этот пакет позволяет подсказать программе-установщику какие разделы и какого размера должны быть созданы жестком диске по умолчанию, а также точки монтирования и предпочтительные файловые системы.
- branding — пакет содержит все графические части, дизайн и, несущественные для сервера, первоначальные пользовательские настройки для оконных сред. Именно здесь можно поменять внешний вид веб-интерфейса, загрузчика, добавить свои картинки для слайд-шоу во время инсталляции.

Методика оценки производительности нереляционных СУБД

Илья Бакулин*

Non-relational databases are now widely used when building scalable web applications. As NoSQL database systems each have its own interface, so failing to choose the right product before beginning to develop application will lead to complete rewrite of database work layer. Most database testing frameworks can not be used to benchmark NoSQL DBMS. We have carried out testing with Yahoo Cloud Serving Berchmark to benchmark non-SQL systems.

В последнее время нереляционные базы данных часто используются при построении веб-приложений, проектируемых с упором на масштабируемость. Это становится своего рода «модой». Но ведущие специалисты-архитекторы, реализующие свои идеи во всемирно известных проектах (Facebook, Twitter, Digg, Riak) сходятся во мнении, что нереляционные базы не могут полностью заменить традиционные системы хранения данных. Большое значение имеет выбор методики обоснования необходимости использования нереляционной базы данных при разработке той или иной части функционала приложения.

Ни одна NoSQL СУБД не поддерживает ставший стандартным в мире реляционных БД язык запросов SQL. Одной из важнейших особенностей SQL является то, что он позволяет разработчику абстрагироваться от внутренней схемы хранения данных в конкретной СУБД. Хотя между диалектами SQL, используемыми в различных реляционных СУБД, имеются определённые различия, в целом для приложения логика работы с БД не меняется при смене СУБД. Напротив, каждая NoSQL-СУБД предоставляет свой интерфейс (API) для взаимодействия, свою схему хранения данных, и в разных решениях они порой очень сильно отличаются. Попытка сменить движок базы данных в процессе разработки может обернуться непредвиденными сложностями, потерей времени, затраченного на проектирование, программирование и тестирование нового решения, и в конечном итоге ущербом для бизнеса компании.

*Москва, Россия

Очевидно, необходимо тщательное тестирование возможных решений, основанных на различных NoSQL-СУБД, перед собственно созданием системы. Тестирование должно давать как можно более полное представление о том, как ведёт себя выбранная СУБД в различных сценариях, типичных для проектируемого веб-приложения.

Одной из систем оценки производительности, умеющих работать с нереляционными базами данных, является написанный в Yahoo программный пакет YCSB (Yahoo Could Serving Benchmark). Поддержка доступа к любой базе данных может быть реализована путём написания адаптера, обеспечивающего элементарные операции CREATE/REPLACE/UPDATE/DELETE. Немаловажным является то, что становится возможным тестирование и реляционных, и нереляционных БД с помощью одних и тех же тестовых сценариев.

Мы использовали пакет YCSB был использован для оценки производительности хранилища данных для SMS-сообщений при выборе между Cassandra и sharded MySQL. Для тестирования сначала была проведена оценка распределения количества запросов на чтение, запись и обновление, после чего написан конфигурационный файл для YCSB. После этого на одной и той же машине были поочерёдно запущены кластеры Cassandra и MySQL и проведено тестирование.

Результаты свидетельствуют о том, что при использовании Cassandra-кластера достигаются более высокие скорости записи в базу, но более низкие скорости чтения, чем на MySQL. Это, в целом, подтверждается и опытом других пользователей. В настоящее время рассматривается возможность тестирования MongoDB, для которой в составе YCSB также имеется адаптер.

Свободные лицензии и российское законодательство: перспективы развития

Сергей Середа*

There is a number of legal issues with FOSS licensing in Russia which in some circumstances could be very serious. Moreover, prepared bill with changes to Civil Code contains initiatives which could only worsen this situation. FOSS enthusiasts in these circumstances should actively participate in debates on developement of the Russian Civil Code to prevent potential and fix actual compatibility problems of FOSS licences.

До относительно недавнего времени в России «свободное» программное обеспечение существовало просто как явление, «независимо» от гражданского законодательства. Но по мере его распространения росло количество конечных пользователей, значительное число разработчиков стали участниками или даже основателями проектов разработки свободных программ, на внутреннем рынке появились сначала модемы и маршрутизаторы, работающие на GNU/Linux, затем стали продаваться телевизоры, видеопроигрыватели и приставки с этой же ОС «на борту», приобрели популярность «умные» телефоны и планшетные ПК на базе ОС Android, развился бизнес по разработке программно-аппаратных решений на базе FOSS и оказанию услуг по их поддержке. Что же касается собственно операционной системы GNU/Linux, то на уровне Правительства она была названа целевой для сферы образования и госсектора в целом.

На этом этапе правовая оценка свободного ПО стала необходимостью. Эта работа была проведена независимо разными исследователями, однако её результаты в целом схожи. В результате анализа текстов свободных лицензий и оценки используемого ими механизма лицензирования в контексте отечественного правового поля можно выделить следующие проблемные моменты:

1. английский язык, на котором написано большинство свободных лицензий, начиная с GNU GPL;
2. отсутствие явного указания территории, на которой действует лицензия;

*Москва, Россия, Движение «ПОтребитель», serge_sereda@hotmail.com

3. отсутствие явного указания срока действия лицензии;
4. отсутствие явного указания суммы авторского вознаграждения (или безвозмездности лицензионного договора);
5. несоблюдение письменной формы сделки при заключении лицензионного договора на свободное ПО;
6. противоречие «вирусного механизма» лицензий на свободное ПО положениям п. 2 ст. 428 ГК РФ, п. 4 ст. 1233 ГК РФ и п. 4 ст. 1260 ГК РФ.
7. эквивалентность безвозмездной лицензии на свободное ПО дарению, запрещённому для юридических лиц;
8. отнесение экономии от бесплатного использования свободного ПО к внереализационным доходам, которые должны облагаться налогом, исчисляемым на основании рыночной стоимости использования аналогичного коммерческого ПО.

Мнения специалистов о перечисленных коллизиях с отечественным законодательством разошлись: одна часть исследователей считает, что, несмотря на проблемные моменты, свободное ПО в России вполне законно (к ним, по большей части, относятся энтузиасты FOSS), другая же их часть считает, что противоречия с российским законодательством слишком серьёзны, чтобы считать свободные лицензии соответствующими Российским законам (этой точки зрения придерживаются «ортодоксальные» юристы).

Следует также отметить, что это расхождение мнений касается лишь программ для ЭВМ и баз данных (для которых допустимы «обёрточные» лицензии). В отношении же свободных лицензий на другие объекты авторского права, в первую очередь — литературные и музыкальные произведения, разногласий практически нет — если подобные лицензионные договоры не заключаются в письменной форме, они (за небольшими исключениями) являются ничтожными на территории РФ.

Теоретически, указанная ситуация может быть разрешена совместными действиями общественных организаций, осуществляющих поддержку экосистемы свободных лицензий, и отечественного законодателя. Что касается общественных организаций, то, например, Creative Commons занимается адаптацией своих лицензий под законодательство РФ, в то время как Free Software Foundation лишь допускает такую возможность. Отечественный же законодатель, как выяснилось, придерживается в этом контексте довольно радикальной точки зрения. Авторы подготовленного пакета поправок в Гражданский Кодекс РФ считают, что несовместимость сво-

бодных лицензий и российского законодательств является системной, и проблему можно решить лишь созданием альтернативного механизма, дающего те же результаты, что и свободные лицензии.

Для этого предлагается ввести новый способ распоряжения исключительным правом — временное предоставление произведения в пользование неограниченному кругу лиц на условиях, определённых правообладателем. Указанный механизм планируется зафиксировать в новой, шестой части статьи 1233 ГК РФ. Однако, чтобы им воспользоваться, правообладателю придётся разместить на официальном сайте федерального органа исполнительной власти по интеллектуальной собственности (`www.fips.ru`) заявление «о предоставлении любым лицам возможности безвозмездно использовать принадлежащий ему результат интеллектуальной деятельности» с указанием условий, на которых предоставляется эта возможность, а также срока действия заявления. При этом предоставлять произведение для использования публике можно только безвозмездно и только если на его использование не заключено ни одного возмездного лицензионного договора. Отказаться от подобного заявления или изменить зафиксированные в нём условия до истечения указанного в нём срока нельзя.

Что касается собственно свободных лицензий, то авторы законопроекта предлагают трактовать их не как лицензионные договоры, а как описанные выше заявления.

Однако, несмотря на озвученную выше позицию авторов поправок, этот же законопроект предполагает и расширение положений об «обёрточных» лицензиях, к которым весьма близки лицензии на свободное ПО. Согласно предлагаемой формулировке новой части 5 статьи 1286 ГК РФ, заменяющей по смыслу часть 3, «обёрточная» лицензия [1] — это лицензионный договор, заключаемый «в упрощенном порядке» (в форме договора присоединения), но являющийся безвозмездным, «если договором не предусмотрено иное». Следует отметить, что, намеренно или нет, но авторы поправок указанной формулировкой устраняют проблему отсутствия явного указания суммы авторского вознаграждения в текстах свободных лицензий. Правда, упрощённый порядок заключения лицензионного договора, к сожалению, так и остался привилегией программ для ЭВМ и баз данных, несмотря на активную торговлю «электронными» эк-

[1] сам этот жаргонный термин в законе, естественно, не используется

земплярами аудиовизуальных и литературных произведений в российской рознице.

Предлагаемые же поправки в текст статьи 1211 ГК РФ скажутся на свободных лицензиях однозначно негативно. Указанная статья определяет, право какой из сторон внешнеэкономической сделки применяется, если в соответствующем договоре об этом ничего не указано. Заключение лицензионного договора (в т.ч. свободной лицензии) с иностранным физическим или юридическим лицом, соответственно, регулируется указанной статьёй ГК. В соответствии с нынешней формулировкой этой статьи к лицензионным договорам, фиксирующим внешнеэкономическую сделку, по умолчанию применяется право страны лицензиара. В контексте существующих коллизий между свободными лицензиями и российским законодательством такая формулировка позволяет считать большую их часть несущественной в случаях, когда в России используется или дорабатывается свободное ПО, изначально созданное за рубежом.

В предлагаемой же новой редакции статьи 1211 ГК РФ, наоборот, говорится, что к лицензионным договорам должно применяться «право страны, на территории которой лицензиату разрешается использование результата интеллектуальной деятельности». Это положение, правда, содержит оговорку о том, что «если такое использование разрешается на территории одновременно нескольких стран» то применяется «право страны, где находится место жительства или основное место деятельности лицензиара», но она, к сожалению, проблемы не решает. Согласно части 3 статьи 1235 ГК РФ, в отсутствие явного указания на территорию, на которой согласно лицензионному договору предоставляется исключительное право, ею считается территория России, а поскольку в свободных лицензиях о территории ничего явно не говорится, то согласно будущей редакции статьи 1211 ГК РФ, к лицензиям на иностранное свободное ПО должно применяться право РФ.

Таким образом, общий эффект от планируемых изменений гражданского законодательства РФ на правовое положение свободных лицензий следует признать отрицательным. В отличие от Европейского союза, не просто признавшего свободные лицензии, но и разработавшего совместимую с ними собственную EUPL, в России лишь создаются дополнительные препятствия для этой «инициативы снизу» по саморегулированию рынка программного обеспечения. Такой подход, при его последовательном применении, приведёт к невозможности использования преимуществ свободного ли-

цензирования российскими разработчиками и конечными пользователями, остановит обмен технологиями в области обработки данных и усилит зависимость России от поставщиков коммерческого программного обеспечения. Также, он прямо противоречит существующим и активно спонсируемым инициативам по внедрению свободного ПО в российском госсекторе, и создаёт существенные препятствия для развития российского инновационного бизнеса в сфере информационных технологий.

По мнению автора, свободное сообщество должно уделить особое внимание анализу планируемых изменений в российском гражданском законодательстве и их последствий для развития свободного ПО, а также принять активное участие в публичном обсуждении законопроекта и внесении в него корректив, которые бы позволили сократить число коллизий свободных лицензий с ГК РФ, вместо того, чтобы их усугублять.

Список литературы

[1] Белопольский Э. Язык ничтожной сделки // Бизнес-адвокат, 1997, № 22. – Режим доступа к электрон. дан.: http://www.juristlib.ru/book_305.html.

[2] Брауде–Золотарев М., Гребнев Г., Протасов П., Ралько А., Сербина Е. Лицензионные договоры и Российское законодательство. – Режим доступа к электрон. дан.: http://vvv.srcc.msu.su/%7Eserbina/INFO-FOSS.RU/Digest3.pdf

[3] Проект ФЗ «О внесении изменений в части первую, вторую, третью и четвертую Гражданского кодекса Российской Федерации, а также в отдельные законодательные акты Российской Федерации». – Режим доступа к электрон. дан.: http://www.economy.gov.ru/minec/activity/sections/corpmanagment/civil_code/full_text_civil_code/140411_gk?presentationtemplate=docHTMLTemplate1&presentationtemplateid=2dd7bc8044687de796f0f7af753c8a7e&WCM_Page.ResetAll=TRUE&CACHE=NONE&CONTENTCACHE=NONE&CONNECTORCACHE=NONE

[4] Середа С.А. Открытое программное обеспечение: проблемы лицензирования и доказательства легальности. – Режим доступа к электрон. дан.: http://consumer.nm.ru/osp_law.htm

[5] Середа С.А. Свободны ли в России «свободные лицензии?» // "Патенты и лицензии". – 2009. N4. – Режим доступа к электрон. дан.: http://consumer.stormway.ru/foss&law.htm

[6] Тяпкина Е. Правовой статус GPL в России // «Компьютерра». – 2002. №13 от 9 апреля. – Режим доступа к электрон. дан.: http://offline.computerra.ru/print/offline/2002/438/17257/.

Виртуализация: история развития и технологии аппаратной поддержки

Павел Круглей[†]

The history of virtualization and virtual machines is briefly discussed. The main types of virtualization are described, with focus on hardware-aided types: the first generation — VT-x, VT-d, AMD-v technologies, the second one — EPT / RVI, and the wave of technology, virtualization on mobile devices.

Истоки (1970е)

Всё начиналось с виртуализации памяти на машинах второго поколения в качестве средства расширения размеров оперативной памяти. Потребность в механизме расширения возникла из-за того, что использовавшаяся в то время память на ферритовых сердечниках стоила чрезвычайно дорого. Поэтому казалось логичным виртуализовать ее, то есть расширить за счет использования внешних устройств.

Впервые виртуальная машина появилась в 1961 году в супервизоре суперкомпьютера Atlas, который был разработан английской компанией Ferranti. В середине 60-х годов она была реализована в проект IBM M44/44X Project и машине IBM 7044.

Следующим шагом в развитии идеи виртуализации стала концепция «виртуальной машины». Она появилась в 1965 году, когда исследователи в корпорации IBM предприняли экспериментальную попытку разделить компьютер на отдельные небольшие части. Это направление исследований привело к созданию многопользовательской операционной среды на машинах IBM System 370 и System 390 и операционной системы VM/ESA, совместно называемых генеалогической линией IBM VM (Virtual Machine).

[†]Минск, Беларусь, nanoflat@gmail.com

Виртуализация в СССР

Проект СВМ/Система Виртуальных Машин являлся частью комплексной программы ЕС ЭВМ (аналога IBM System/370), которая была начата в 1969 году. Адаптация системы IBM VM/370 Release 5 и программных продуктов ее окружения были основными целями в начале проекта СВМ.

СВМ (VM, и её ранняя версия CP/CMS) — первая система, в которой была реализована технология виртуальных машин. Виртуализация в СВМ была последовательной и полной, в частности, на виртуальной машине можно было запустить другую копию системы СВМ, и так далее.

Архитектурно СВМ состояла из нескольких независимых компонентов. Центральным компонентом был монитор виртуальных машин (МВМ, IBM-овское название — CP, Control Program), который управлял аппаратурой реальной ЭВМ и реализовывал набор виртуальных машин с заданной конфигурацией. Остальные компоненты представляли собой операционные системы или системонезависимые программы виртуальных машин, работавшие под управлением МВМ: подсистема диалоговой обработки (ПДО), подсистема сетевой передачи файлов (ПСП), подсистема логической коммутации абонентских пунктов (ПЛК), подсистема анализа дампов (ПАД), подсистема дистанционной передачи файлов (ПДП), подсистема контроля технических средств (ПКТ), средства генерации и обслуживания (СГО).

Типы виртуализации

Обычно среди решений на основе виртуализации выделяют:

- Эмуляция аппаратуры — в хост-системе создается виртуальная машина, которая моделирует какую-то другую аппаратную архитектуру.
- Полная виртуализация — использует виртуальную машину (гипервизор), которая выступает как посредник между гостевой операционной системой и реальным оборудованием. Некоторые инструкции защищенного режима должны перехватываться и обрабатываться внутри гипервизора, поскольку аппаратура не доступна непосредственно из операционных систем, доступ к ней предоставляется через гипервизор.

– Паравиртуализация — требует модификации гостевой операционной системы, что является недостатком. Однако в этом случае обеспечивается производительность, близкая к производительности невиртуализированной системы.
– Виртуализация уровня операционной системы — операционная система одна и просто изолируются один от другого сервера, работающие под ее управлением.

Виртуализация в x86

Впервые апаратная виртуализация была реализована в 386-х процессорах и носила название V86 mode. Этот режим позволял запускать параллельно несколько DOS-приложений.

В 2005-м году компании Intel и AMD представили решения аппаратной поддержки виртуализации — INTEL VT и AMD-V. Были введены дополнительные инструкции для предоставления прямого доступа к ресурсам процессора из гостевых систем. Этот набор дополнительных инструкций носит название Virtual Machine Extensions (VMX). VMX предоставляет инструкции: VMPTRLD, VMPTRST, VMCLEAR, VMREAD, VMREAD, VMWRITE, VMCALL, VMLAUNCH, VMRESUME, VMXON и VMXOFF.

Архитектура AMDV похожа на VT и предоставляет те же функциональные возможности, однако предусматривает также ряд дополнительных функций, отсутствующих у Intel VT.

Технология VT-d позволяет избежать полной виртуализации устройств ввода-вывода. Используя VT-d, VMM сможет «прикреплять» драйверы физических устройств к ВМ, что позволит гостевой ОС взаимодействовать с прикрепленным устройством без передачи управления VMM посредством механизма DMA (прямого доступа устройств к памяти минуя процессор).

Следующее поколение аппаратной виртуализации предусматривает виртуализацию памяти. Это технологии AMD NPT (Nested Page Tables) / Intel EPT (Extended Page Tables). В целом технологии HAP (Hardware Assisted Paging), NPT, EPT и RVI (Rapid Virtualization Indexing) — скорее изобретение маркетологов, так как на самом деле обозначают одно и то же. При вызове операций из гостевой ОС, она отдает команду VMLAUNCH и начинает исполнять свой код, пользуясь далее инструкцией VMRESUME. Однако при операциях с памятью, гостевая ОС должна работать через мони-

тор виртуальных машин VMM для управления памятью (механизм называется Shadow Paging). С использованием же Extended Page Tables (EPT) — гостевая ОС может сама управляться со страницами памяти, включая контроль Page Faults, которые, кстати, проходят через гипервизор и вызывают VMEXIT. Инструкция VMEXIT — это по сути передача управления Монитору виртуальных машин гипервизора, от которого ожидаются какие-то действия. Соответственно, чем меньше вызовов VMEXIT — тем лучше.

Мобильная виртуализация

Компания VMware продемонстрировала MVP (Mobile Virtualization Platform), гипервизор для мобильных устройств, позволяющий одновременно использовать на телефоне такие платформы, как Google Android и Windows Mobile. Разработку планируется использовать для создания на мобильном устройстве дополнительного защищенного окружения для работы с приватными данными.

Гипервизор MVP был продемонстрирован на планшетном ПК Nokia N800. 128MB ОЗУ и 256MB Flash оказалось достаточно для одновременного запуска двух виртуальных машин с Windows CE 6.0 и Google Android.

TRANGO — гипервизор, а точнее виртуальный процессор, весящий 20 Kb, позволяющий запускать на мобильном устройстве с RISC-процессором от ARM или MIPS несколько гостевых ОС: eCos, Linux, Windows CE и другие ОС.

Компания Citrix объединила свои усилия с группой разработчиков Open Kernel Lab (OK Lab), создав уникальную мобильную платформу под рабочим названием Nirvana Phone.

Проект Nirvana Phone можно рассматривать как гибрид карманного смартфона и традиционных настольных гипервизоров. Виртуальный рабочий стол XenDesktop внутри коммуникатора Nirvana Phone легко разворачивается в полноценное рабочее место с большим монитором, мышью и клавиатурой за счет применения мобильного клиентского модуля виртуализации OKL4 Microvisor 4.0. Сами разработчики считают, что оптимальным способом подключения периферийных устройств будет беспроводной интерфейс Bluetooth — использование этого канала позволяет применять массу уже выпускаемых серийно устройств.

Merge-a-fork или скрестим вилки

Михаил Шигорин*

Source code availability might tempt to «just make a copy of it». This might lead to fragmentation which in its turn is capable of splintering the efforts, introducing incompatibilities, and even development stall — but doing it properly may also heavily help with the feature diversity. Questions to discuss are what's a fork and a merge, the price of divergence and convergence, and joint competition.

Когда мы что-то делаем, то нередко основываемся на уже существующем и дополняем или дорабатываем его. При этом вне зависимости от того, код это, документация или конфигурация — происходит фактическое «раздвоение» объекта. И если уже существующее продолжает развиваться, то это раздвоение называется «форк». Если такие производные варианты сливаются полностью или частично, постоянно или периодически — такое объединение называется «мерж».

Форк чреват тем, что либо усилия на развитие в основе схожих вещей будут дублироваться (без малого худший случай), либо потребуются дополнительные решимость, время и силы на периодическое «втягивание» наработок коллег, либо же произойдёт загнивание менее продуктивной ветки вместе с уникальными для неё наработками.

С другой стороны, ожидаемый форк (когда событие ответвления с целью дальнейшей разработки является ожидаемым и приветствуемым) может оказаться весьма мощным средством проверки самых различных идей реализацией: тогда наиболее удачные затем прямо или же после переработки (как правило, с учётом возникших замечаний коллег и параллельно происшедших изменений в затрагиваемой части) мержатся в более майнстримную ветку.

Мерж чреват в первую очередь неизбежной затратой времени — как на собственно техническую часть вопроса, так и на предварительное согласование с последующей утруской и усушкой вновь образовавшихся проблем.

*Киев, Украина, mike@altlinux.org

Хорош же мерж тем, что уменьшение объёма разницы между развивающимися параллельно ветками приводит к уменьшению латентных затрат времени на отслеживание и втягивание интересных наработок коллег.

В докладе рассматриваются типичные виды, причины и последствия форков (причём всегда есть что добавить по опыту аудитории), а также более-менее соответствующие варианты мержей.

The simple programming for students to process the data at laboratory works

Sofiya Apunevych*, Stepan Apunevych†

The data obtained during laboratory lessons by physics course students can be processed in a number of different ways, using quite different tools, ranging from calculator to computer algebra systems (eg Matlab, GNU Octave or even GNU R). However, we argue that this procedure would have the most pedagogical effect when done personally by the student, programming the task on his or her own, running program and getting the results. This way brings more knowledge to the student, since it also introduce him or her to the world of computations, giving the very sense of idea. Also, it reminds the people the direct purpose of computers — that is, to crunch numbers — this primary function nowadays is barely remembered.

The questions rise, how to implement such approach: 1. What programming language to use? 2. Which of numerous development environments to use?

The answers we propose are following. 1. C programming language. 2. Free software (Linux), the IDE Kuzya, developed by Programming Lviv Linux User Group community (`http://www.pllug.totalh.com/`).

Why C? Pros are obvious: C is the eldest and most wide-spread of programming languages. Its syntax was inherited by most of popular general-purpose programming languages, C++, Java, Perl/PHP, C# to name the least. Its fundamentals had shaped the face of modern IT systems. The cons are drawbacks of C language, as concerned to novices, its proximity to hardware in some aspects and excessive flexibility, posing the problems for non-professionals. These difficulties can be overcomed with use of some libraries and styles of programming. We have used the textbook 'Numerical Recipes. Art of Scientific Computing' (by Press et al., 1992), available online at `www.nr.com`. This book solves both problems. First, it proposes very clean and concise yet comprehensive treatment of C language, in as much as 28 pages of 'Preliminaries'. The teacher can use it as an affordable reference

*Lviv, Ukraine, Lviv National University of Veterinary Medicine and Biotechnologies named after S.Z. Gzhytskyj, `sofiya.apunevych@gmail.com`

†Lviv, Ukraine, Lviv Linux Users Group, `apusbird@gmail.com`

for students (unofficial translation). On the other hand, 'Numerical Recipes' provides a library, a full set of functions on numerical methods, the framework for vectors, matrices.

Why IDE Kuzya? Free (Open Source) Software provides the wide range of development tools (compilers, editors, integrated development environments). The distributions of the software are highly available, can be easily installed (eg Ubuntu). Some of this software is cross-platform. But, anyway, these powerful tools for professionals require quite a lot of learning and training. The IDE Kuzya is aimed precisely to address the problem. It is developed mostly by students, and simplifies the learning curve for novices. It runs seamlessly on MS Windows, Linux and Mac OS, and perfectly fits the needs of pupils, having the clean user interface, and lowering the demands to user.

To summarize, the student is left with computer. The tasks are the following: to type the measurements data into the data file and save it; to open the program of statistical description of data (Ch. 16 of NR) with Kuzya and modify it for these data, to run and gain results. Its really affordable task to cope by any student, it develops basic skills and gives the pupil the feeling of simplicity. For more advanced students the same task can be elaborated.

Conclusions

The Kuzya IDE together with 'Numerical Recipes' is a powerful teaching tool. A student can grasp the essence of scientific computations during the physics course.

Organic software. Как это работает

Константин Лепихов*

The 'organic software' term was declared by Mozilla for their software. Its main idea is patent and blob-free software that respects the user. But it also respects open standards, which create a level playing field for any individual, company or organization to create Web content for others. And it is a manifestation of Mozilla's core belief in the importance of providing vehicles for participation on the Internet (`http://www.mozilla.org/about/mozilla-manifesto.html`).

Концепция «органического» программного обеспечения была рождена в недрах Mozilla как способ существовать на рынке закрытых программных решений и вести успешную конкуренцию с ними. Главная идея подхода — программное обеспечение, свободное от патентов и встроенных бинарных объектов, которое с уважением относится к пользователю. Однако кроме пользователя, уважительное отношение задекларировано также к открытым стандартам, дающим пространство для того, чтобы любой индивидуум, компания или организация могли создавать для других вебконтент. Это проявление веры Mozilla в важность предоставления средств для участия в развитии Интернет. Ключевая идея изложена в `http://www.mozilla.org/about/mozilla-manifesto.html`.

Рассмотрим современный рынок браузеров и мобильных платформ. Инфраструктура Mozilla является в полной мере и платформой для разработки, нет только вендора, который сможет воплотить идеи в конечный продукт, будь то телефон или MID.

Microsoft развивает платформу WM7 и браузер IE. Подход рассматривает технологии всего лишь повод увеличить сумму для заказчика. Потенциал платформы WM7 пока не очень неясен. Браузер является аутсайдером по реализации стандартов и чемпионом по количеству ошибок проектирования и скрытых дефектов.

Opera стремится реализовать все первой. Реализация может быть далека от идеала и в ущерб другим более полезным вещам, но должна появиться на месяц раньше конкурентов (оставим за рамками и код браузера, поскольку нам его никогда не покажут).

*Mozilla.Россия

Google Android/Chrome создается компанией, у которой есть платформа, копирующая все самое лучшее, что есть в iOS, WM или Symbian, но открытая для разработки и допускающая возможность создания закрытых форков. Есть движок рендеринга, который также открыт и допускает возможность создания закрытых форков. По отзывам, в создании этого движка участвовали люди из Apple и Symbian. У компании есть собственный браузер, который очень быстр и очень хорош. Но, к сожалению, бранч с кодом браузера компании закрыт, а в ОС много блобов и DRM.

Mozilla Foundation преследует совсем другие цели: у сообщества не может быть конкурентов, поскольку сообщество ничем не торгует. Но у него может быть PR, чтобы быть услышанным на рынке коммерческих продуктов. И у сообщества есть инфраструктура, чтобы разработка самого лучшего и удобного для пользователя бразуера ничем не стесняла разработчика.

Основные компоненты инфраструктуры разработки:

- Система отслеживания ошибок (Bugzilla)
- SDK и публичные репозитарии (Jetpack и Incubator)
- Документация на языках, доступных разработчикам любого уровня (MDN и mozhacks)
- Каналы поддержки (irc/web/social media)
- Система QA сборок и автоматического тестирования
- Правовая поддержка и защита интеллектуальных прав.

«Open Source Бизнес» или о том как открыть 90% и остаться со штанами

Руслан Закиров*

The topic of the report is combining business with open source products. The personal experience of working in company producing mostly free / open source products is represented in several field-tested principles.

Данный рассказ — о том, как можно совместить бизнес с открытыми продуктами. В его основе нет никакой книжной теории, а исключительно личный опыт: более 5 лет автор работает в небольшой бостонской компании, Best Practical Solutions, 90% продуктов которой распространяются под открытыми лицензиями. Ниже приведены основные принципы, примененные и доказавшие на практике, что построенный на них бизнес может быть успешным.

Флагманский продукт под открытой лицензией

Этим продуктом является Request tracker (RT) — система обработки заявок пользователей (ticket tracking). Продукт приносит основной доход компании. В чем его успех? Мы не пытались создать что-то, что заработает для всех и сразу. Вместо этого мы создаем программу, которую можно настроить, расширить и интегрировать с другими решениями. Отсутствие границ для применения увеличивает количество возможных пользователей. Каждый новый пользователь — потенциальный клиент. Продукт ориентирован на бизнес: так сложилось, что бизнес готов платить за инструменты, когда они приносят прибыль или снижают расходы.

Плюсы и минусы Open Source

Для клиентов основной плюс заключается в цене продукта. При этом наш продукт не является полностью бесплатным. В какой-то момент клиенту не хватит функционала «из коробки», и тогда ему приходится потратить деньги. Но клиент сам решает когда, на что

*Москва, Россия

и сколько потратить. В случае открытых программ выбор гораздо шире. Однако, у бизнеса есть свои опасения на счет открытых продуктов. Одно из самых главных опасений — что завтра, когда понадобиться помощь, никого не окажется рядом. Поэтому существуют такие компании, как Best Practical Solutions.

Роль сообщества

Сообщество пользователей вашего продукта — это ваши потенциальные клиенты, т. е. самое ценное, что у вас есть. Поэтому сообщество нужно холить и лелеять, а когда его еще нет, его нужно взращивать. Хорошее сообщество приносит не только клиентов, но и маркетологов, которые будут рекламировать продукт, распространителей, службу бесплатной поддержки, авторов технической документации, специалистов для консультации, разработчиков.

Опыт велосипедов

Если у вас классическая закрытая компания, то ничего плохого в этом нет, но это не значит, что вы не можете заработать дополнительные деньги на открытии ряда технологий. Многие компании разрабатывают вспомогательные инструменты внутри, и часто это «велосипеды», возникшие, когда доступные решения не заработали, и пришлось написать свое. Если это заработало для вас, то возможно заработает и для других. В Best Practical разрабатывали SVK — еще одну систему контроля версий, которая решала ряд внутренних задач. В итоге к разработке присоединилось 28 разработчиков, и пришло несколько клиентов, которых мы и не ожидали.

Заключение

Определенно, опыт показывает, что компания может существовать, разрабатывая только программы с открытым исходным кодом. Открытые решения дают ряд финансовых преимуществ пользователям, что делает их привлекательными для бизнеса. Сообщество пользователей становится важным фактором финансового успеха продукта. Регулярные инвестиции в построение сообщества — насущная необходимость. Открытый код может стать дополнительным источником дохода от вспомогательных технологий, разрабатываемых внутри закрытых компаний.

Построение домашнего медиацентра с помощью LIRC

Алексей Бутько*

LIRC is a package that allows to decode and send infra-red signals of many (but not all) commonly used remote controls. All remote controls that are supported by learning remote controls, i.e. almost any, should also work with LIRC. The most important part of LIRC is the lircd daemon that decodes IR signals received by the device drivers and provides the information on a socket. The user-space applications allow you to control your computer with remote control, i.e. to send X events to applications, start programs and much more on just one button press.

Для многих компьютер заменил все аудио и видео-устройства в доме. У многих представителей профессии, включая автора, в доме отсутствует даже телевизор. Однако, общение с компьютером как с медиацентром происходит совсем иначе, чем с бытовыми медиа-устройствами. Хоть «компьютерный» способ привычен и понятен, он все же не очень удобен.

Удобнее использовать для управления медиацентром пульт. Проблему частично решают беспроводные мыши и клавиатуры, но они громоздки.

В Linux имеется мощный инструмент работы с ПДУ — lirc (Linux Infrared Remote Control). В первую очередь для работы с пультом требуется ИК-приемник. Подойдет практически любой ИК-порт или TV-тюнер (список поддерживаемых устройств можно найти на сайте проекта `http://lirc.org/html/table.html`). Однако, если такого устройства нет под рукой, нет необходимости его покупать. ИК-приемник, позволяющий работать с любым пультом, можно изготовить: для этого требуется инфракрасный датчик от телевизора или любой подобный, расчитанный на напряжение 5 вольт.

Такое решение является наиболее простым и доступным, однако, т. к. COM-порт — уже исчезающий вид интерфейса, более привлекательными выглядят USB-приемники. Для его изготовления потребуется микроконтроллер.

*Минск, Беларусь, `gnomkun@gmail.com`

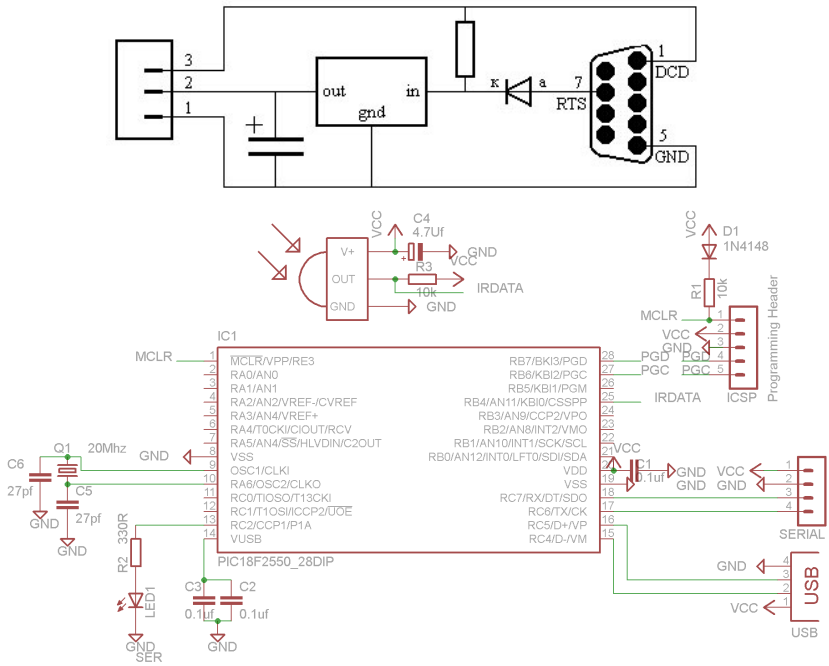


Рис. 7. Схема инфракрасного приемника

Приемник базируется на микроконтроллере PIC18F2455, который может работать с USB-портом и является меньшей и более дешевой версией 18F2550. Семейство 18F можно программировать при помощи универсального PIC-программатора.

В состав пакета LIRC входят:

– драйверы различных устройств (модули ядра)
– демон lircd, преобразующий ИК сигналы, полученные от драйвера, в стандартные сообщения, которые прикладные программы могут получить через сокет
– демон lircmd, получающий сообщения от lircd и имитирующий мышь в X Windows
– irexec — запуск программ по нажатию кнопки ДУ
– irxevent — посылка X Windows сообщения по нажатию кнопки ДУ

– irpty — псевдотерминал, запускающий программу и имитирующий нажатие клавиш клавиатуры

Кроме того, стоит упомянуть вспомогательные программы для отладки и настройки:

– irrecord — утилита для записи сегналов пульта и создания lircd.conf
– irw — читает сообщения с сокета lircd и выдает на stdout; в качестве параметра можно указать имя сокета
– ircat — отладочная программа для конфигурационного файла /.lircrc; в качестве параметра указывается имя программы (точнее имя описывающей её секции); по нажатию кнопки на пульте ДУ ircat выводит на stdout строку, привязанную к этой кнопке
– mode2, smode2, xmode2 — осциллоскоп для инфракрасных сигналов
– irsend — посылает команды на инфракрасные приемники (если позволяет оборудование)

В Linux существует множество приложений, направленных на централизованную работу с мультимедиа, например MythTV, Elisa, XBMC. Однако, очевидно, что пульт может использоваться совместно с множеством других программных пакетов.

GRID-технологии в исследованиях дымовой плазмы

Сподарец Д.В., Драган Г.С.*

The paper describes the use of GRID systems and computer clusters for smoky plasma research. Offers insight into the experience of expanding the functionality of TERM software and hardware complex for GRID systems.

В экспериментах все чаще используют вычислительные кластеры и GRID-системы. Связано это с тем, что количество информации, полученное в следствии проведения эксперимента с использованием современного измерительного оборудования, достаточно велико. Немаловажным фактом, который играет роль при выборе суперкомпьютеров для автоматизации исследований и обработки результатов, является уменьшение ошибки измерений, а также возможность использовать новые технологии диагностики, например, позволяющие наблюдать процессы в реальном времени.

Основной объект наших исследований — дымовая плазма. Она является разновидностью низкотемпературной термической плазмы с конденсированной дисперсной фазой, образуется в продуктах сгорания топлива и состоит из газовой фазы с легкоионизируемой примесью атомов щелочного металла и конденсированной в виде частиц оксидов металлов и сажи [1].

Для проведения диагностики таких параметров дымовой плазмы, как температура газовой и конденсированной фаз, концентрации электронов и атомов, нами был разработан аппаратно-программный комплекс TERM [2]. В его основе лежат оптические методы диагностики плазмы, а программная часть построена на свободном программном обеспечении. Среди новых возможностей комплекса, которые уже реализованы или находятся на стадии реализации, можно отметить дополнение его собственным вычислительным кластером, адаптация для работы в GRID-системах, разработка механизмов проведения виртуальных экспериментов [3].

Комплекс TERM работает следующим образом:

*Одесса, Украина, Одесский национальный университет имени И.И.Мечникова, m31@root-ua.com

1. Экспериментально определяется температура газовой и конденсированной фаз, а также концентрация электронов и атомов в плазме.
2. На основе теории дымовой плазмы и измеренных экспериментально параметров проводится расчёт заряда частиц, а также определяется функция распределения частиц по заряду.
3. Полученные данные зарядов используются для определения пространственного концентрационного неравновесия и сил диффузионно-дрейфового давления свободных носителей заряда на частицы. Основой для данных расчётов служит теория обобщённого потенциала.
4. Значения сил диффузионно-дрейфового давления учитываются при расчёте парного взаимодействия частиц, а также в уравнениях молекулярной динамики для дымовой плазмы.
5. Результатом работы комплекса является моделирование 3D структур заряженных конденсированных частичек в плазме. Этот этап проводится с использованием GRID Украины.

Базовой ОС комплекса является ALT Linux. Для захвата и обработки видеопотоков применяется библиотека компьютерного зрения OpenCV. В качестве инструментария, при проведении расчётов, задействованы Octave и Mathematica. Разработан ряд собственных специфических алгоритмов, например, для проведения анализа спектров излучения плазмы.

При совмещении реального физического эксперимента и виртуального моделирования на базе GRID Украины, стало возможным более глубокое изучение влияния объёмных и поверхностных процессов, происходящих в области пространственного заряда конденсированной частицы, на кинетику и динамику дымовой плазмы.

Список литературы

[1] Dragan G.S., Malgota A.A. et al. // Proc. Sc. and Tech. Meet., Alma-Ata, USSR, October 25-31, 1982. – Inst. High Temp., Acad. Sci. of the USSR, 1984. – P. 191 – 192.

[2] http://term.m31.org.ua

[3] Spodarets D.V., Dragan G.S. Method of Experimental Research of Long-Range Interactions in Smoky Plasmas // Book of Abstr. of the ICPDP 2011, Garmasch-Partenkirchen. Germany.: 2011. P. 74.

Свободное производство информации в постиндустриальном обществе

Елена Иванова*, Дмитрий Костюк†

The concept of postindustrial society is discussed in relation to free / open source software community-driven production. Its better correlation with Bell principles than classic information-as-a-good approach is shown.

Концепция постиндустриализма в качестве следующей стадии социального развития приобрела широкую известность в 1970-х годах, когда Д. Беллом были сформулированы следующие положения [1], лежащие в его основе: главенствующая роль знаний в обществе (1), главенствующая роль производства не товаров, но услуг (2), университет как очаг производства новых знаний и инноваций (3). Исходно реализацию перечисленных принципов затрудняла дороговизна и малая эффективность коммуникации между специалистами. Ставшая популярной в 90-х концепция информационного общества [2] включила в себя построение глобального информационного пространства с использованием новых компьютерных технологий, и таким образом в значительной степени снизила коммуникационный барьер за счет на порядок более эффективного удаленного взаимодействия.

Популярное толкование положения о информации и знаниях, как главном продукте производства, рассматривает поточное производство информации как товара в рамках сервисной (т. е. ориентированной на рынок услуг) экономики. В определенном смысле первый и третий из сформулированных Беллом признаков постиндустриального общества рассматриваются в качестве сателлитов положения о примате производства услуг.

Производство информации более прибыльно в сравнении с материальным производством, так как достаточно изготовить первоначальный образец, а затраты на копирование несущественны. Но эти же свойства информации делают ее неудобным товаром с точки

*Москва, Россия, http://coofeed.com
†Брест, Беларусь, dmitriykostiuk@gmail.com

зрения традиционных экономических отношений. Когда под предоставлением услуг понимается торговля информацией или сдача информации в аренду, приходится изобретать искусственные ограничительные механизмы. Необходима развитая юридическая защита прав интеллектуальной собственности. Права на информацию, которые подлежат юридической защите, должны носить монопольный характер (это является не только необходимым условием для превращения информации в товар, но и позволяет извлекать монопольную прибыль, увеличивая рентабельность постиндустриальной экономики). Необходимо огромное количество потребителей, которые готовы платить за информацию, предоставляемую в данной ограниченной форме.

Однако возрастание числа людей, занятых информационными технологиями, коммуникациями и производством информационных продуктов и услуг — людей, основным «средством производства» которых является не оборудование работодателя, а их собственная квалификация [2] — вносит в данный подход коррективы. В результате, существенная часть членов общества оказывается неудовлетворенной накладываемыми на информационный продукт искусственными ограничениями, делающими его использование менее удобным и осознает, что сверхприбыли поставщиков информационного продукта противоречат интересам их, как потребителей. Такая группа имеет средства производства для коллективного воссоздания аналога и сетевую инфраструктуру, позволяющую добиться предсказанной апологетами постиндустриализма быстрой самоорганизации творческих коллективов для решения конкретных задач, а ее представители готовы воспринимать в качестве опосредованного источника прибыли дополнительные факторы, такие как рост личного профессионализма и профессиональную репутацию.

Наиболее ярко на сегодняшний день данное явление проявляется в сфере свободного программного обеспечения. Группы специалистов, неудовлетворенные ситуацией на рынке, объединяются, выделяя часть своего свободного времени (являющегося в постиндустриальном обществе оцениваемым ресурсом) для создания равноценного программного продукта, свободного от искусственных ограничений, связанных с интеллектуальной монополией [3] какого-либо производителя. В качестве страховки интересов потребителей такой продукт часто оснащается собственными лицензионными ограничениями, направленными против его возможной монополизации. В рамках сформулированной концепции информационного

общества данное явление можно рассматривать как массовое коллективное инвестирование, базирующееся на совладении материальными и нематериальными активами (crowd funding). Характерно также, что первоначальными источниками появления свободных программ являлись университеты, в полном соответствии с одним из сформулированных Беллом положений.

Модели бизнеса, построенные вокруг таких продуктов, хорошо укладываются в постиндустриальную модель, предоставляя коммерческие услуги по сопровождению и/или доработке продукта, а также доступ к продукту (консолидированному и обслуживаемому централизованно на мощных вычислительных станциях) в рамках облачных вычислений и технологии Software-As-Service (приложение как услуга).

Явление коллективного производства свободно распространяемых информационных продуктов претерпевает некоторые эволюционные изменения, связанные с большей доступностью для потребителя и проникновением в смежные области, примером чему служат коллективное создание высококачественного энциклопедического контента википедии и родственных проектов, а также активизация на рынке средств мобильной связи. В частности, идеи объединения функциональности сотового телефона и карманного персонального компьютера появились практически сразу после появления первых карманных персональных компьютеров в начале 90-х. Наличие полнофункциональной операционной системы делает смартфоны и коммуникаторы более привлекательными в глазах большинства пользователей. Современные телефоны прекрасно справляются со многими задачами, выходящими за рамки телефонных: работа с электронной почтой, просмотр текстовых документов и электронных таблиц, работа с планировщиком задач и многими другими. Однако резкий рост популярности смартфонов произошел одновременно с развитием инфраструктуры открытой операционной системы Android [4]. Аналогичным по своей сути направлением эволюции в компьютерных технологиях является децентрализация, примером которой могут служить файловый обмен (торенты), электронные системы платежей, такие как BitCoin [5], распределенные социальные сети.

Список литературы

[1] Д. Белл. Грядущее постиндустриальное общество. М., Академия, 1999.

[2] Постиндустриальное общество. http://ru.wikipedia.org

[3] Л. Лессиг. Свободная культура. http://www.gumer.info/bibliotek_Buks/Culture/lessig/index.php

[4] Source: Gartner (May 2011) http://www.gartner.com/it/page.jsp?id=1689814

[5] BitCoin — анархическая пиринговая криптовалюта. http://blogerator.ru/page/bitcoin

Протокол IF-MAP

Олег Орел*

The Interface for Metadata Access Points (IF-MAP) is an open standard client/server protocol developed as one of the core protocols of the Trusted Network Connect (TNC) open architecture. IF-MAP provides a common interface between the database server acting as a clearinghouse for information about security events and objects, and other elements of the TNC architecture.

TNC (Trusted Network Connect) — архитектура, описывающая возможную реализацию подхода к сетевой компьютерной безопасности, унифицирующего решения по обеспечению безопасности на конечных узлах (такие как антивирусное ПО, системы обнаружения вторжений и т. д.), пользовательскую аутентификацию, элементы обеспечения безопасности. Компьютер, подключившийся в сеть, получает уровень доступа к ресурсам по результатам анализа таких его параметров, как уровень защищенности от вредоносных программ, конфигурации, роли пользователя, обновлений ОС. По любому параметру доступ может быть разграничен: например, пользователь, чей компьютер не обладает свежими антивирусными базами, не будет допущен в Интернет. IF-MAP — открытый стандарт, описывающий клиент-серверный протокол обмена данных между элементами (MAP) в TNC-архитектуре. IF-MAP сервер — это централизованное хранилище метаданных (например, о состоянии узлов сети, пользователях и т. д.), предоставляющее механизмы для публикации данных, поиска и подписок всем заинтересованным MAP-клиентам. Модель данных, предусмотренная для IF-MAP сервиса, представляет собой граф, вершины которого — идентификаторы (device, ip-address, mac-address, access-request, . . .), а ребра — метаданные. Например, метаданные типа ip-mac, которые публикует MAP-клиент, работающий совместно с DHCP сервером, соединяют идентификаторы типа ip-address и mac-address (DHCP lease) и содержат дополнительную информацию (время действия адреса). MAP-клиентам доступен поиск на этом графе, а система подписок на изменения позволяет выполнять отложенный поиск.

*Минск, Беларусь, EPAM Systems, Aleh_Arol@epam.com

Голос спонсора: SaM Solutions

Компания SaM Solutions выступает в роли системо-образующего спонсора конференции Linux Vacation Eastern Europe, с момента зарождения LVEE в 2005 году и на протяжении всех лет её проведения.

Сложившаяся корпоративная практика не случайна. Продукты и решения, задействующие Linux и другие Free/Open Source Software проекты, составляют заметную часть пакета разработок SaM Solutions. Кадровая политика компании направлена на поощрение профессионального развития своих сотрудников, организацию их эффективного отдыха и привлечение хорошо мотивированных кандидатов к работе на компанию. Формат конференции LVEE успешно позволяет решать все три задачи.

Одним из направлений нашей деятельности является направление Linux/Unix разработок. Специалисты компании на протяжении десятилетий работают с СПО. Компанией реализован ряд проектов по адаптации ОС GNU/Linux для работы в различных устройствах, построенных на таких платформах как ARM, PowerPC, x86, MIPS. В последние годы — на ведущие позиции выходит разработка управляющего ПО для серверов Enterprise-класса, от низкоуровнего BMC Firmware на основе Linux до высокоуровневых систем контроля виртуализации и графических интерфейсов управления. Надёжность, качество и широкий функционал множества свободных проектов позволяет строить нам системы любого уровня и сложности, опираясь на высококачественные готовые компоненты.

Мы разрабатываем, модифицируем и адаптируем различное свободное програмное обеспечение для наших заказчиков, но не забываем о нём и для своих собственных нужд — наши сотрудники используют в своей работе существующие програмные продукты и вносят вклад в их развитие. Часть внутренней инфраструктуры, а именно интранет-сеть компании, тестовые стенды отдела контроля качества, рабочие места сотрудников профильных подразделений — также работает под управлением СПО (серверные и десктопные платформы GNU/Linux и FreeBSD).

По указанным причинам компания заинтересована в развитии движения open source, в продвижении свободных программных продуктов, и участие в конференции разработчиков и пользователей СПО.

В рамках Unix/Linux направления успешно выполнены проекты для таких знаковых заказчиков, как Novell/SUSE, Fujitsu Technology Solutions и осуществляется партнёрство с компаниями IBM и Oracle/Sun в области Open Source решений.

Линукс-отдел компании SaM Solutions (Dept6/DDC3) уже более десяти лет является настоящей неофициальной кузницей кадров для специалистов в области Free/Open Source Software (FOSS). Код наших сотрудников есть в таких известных свободных проектах как Linux Kernel, Debian GNU/Linux, Alt Linux, Samba, wxWidgets, языке Ruby и его библиотеках и многих других. И SaM Solutions продолжает двигаться вперёд, не останавливаясь на достигнутом.

Голос спонсора: EPAM Systems

Компания EPAM Systems не первый год является спонсорами международной конференции разработчиков и пользователей свободного программного обеспечения LVEE (Linux Vacation / Eastern Europe). Этот год также не стал исключением. Пожалуй, LVEE является самым значимым событием для русскоязычных разработчиков и тестировщиков Open Source. Каждое лето здесь встречаются начинающие специалисты и «ветераны»-разработчики из порядка 10 стран для обмена опытом и общения на профессиональные темы. Наши специалисты также активно участвуют в данной конференции: в качестве докладчиков и организаторов/волонтёров. Это уникальная в своем роде конференция, и именно по этому EPAM Systems очередной раз принимает участие в LVEE в качестве спонсора.

EPAM Systems — одна из крупнейших компаний-поставщиков услуг в области разработки программного обеспечения и решений на территории СНГ и Центральной и Восточной Европе. Созданная в 1993 году, сегодня она имеет 17 представительств в 8 странах мира, в штате работают более 6 тыс. сотрудников, из которых 2,5 тыс. — в Беларуси. Рост компании обеспечивается засчет собственных обучающих программ и передаче опыта от больших специалистов до начинающих разработчиков. Компания EPAM Systems выполняет проекты более чем в 30 странах мира. Основные направления деятельности: разработка, тестирование, сопровождение и поддержка заказного программного обеспечения и бизнес-приложений, а также ИТ-консалтинг с учетом отраслевой специфики бизнеса.

Наша компания участвует в проектах с такими крупными, хорошо известными заказчиками как Google, Novell, Infoblox, Parallels и др. так и с небольшими Neterion, Viaclix, WinBox, в том числе и начинающими свой путь в софтверном бизнесе.

К примеру, для Infoblox была реализована связка между WebUI с BIND и DHCP. Для этого был разработан комплекс решений под управлением Shell и Python скриптов, а также механизм позволяющий вносить правки в BIND и DHCP на языке C. Также был разработан развернутый функционал, автоматизирующий инсталляцию новых устройств и их эксплуатацию, что позволяет значительно упростить управление данными. Встроенный WEB интерфейс позволяет разворачивать, управлять сервисами DNS, DNSSEC, DHCP,

IPAM, устанавливать новые версии ПО, архивировать и восстанавливать из архивов необходимые данные, восстанавливать их после аварии, проводить мониторинг сети и создавать отчеты без необходимости обращения к командной строке.

Еще одним решением реализованным для компании Infoblox являлся программный продукт позволяющий контролировать сетевые изменения, таким образом облегчая идентификацию трудноуловимых проблем конфигурации и соответствие требованиям. Вместо того, чтобы просто регистрировать изменения, система использует внесенную информацию для проверки, анализа и автоматической обработке сетевых изменений. Благодаря инновационной квалифицированной глубокой технике логического анализа, программа изолирует проблемы исправности и конфигурации до того, как они могут вызвать более серьезные проблемы.

Разработанная для анализа сложных сетей система изучает сеть, собирает ключевую информацию, применяет встроенную технику логического анализа и создает оценку исправности сети и список проблем, требующих принятие мер для улучшения качества работы сети.

Правильное использование свободного ПО в разработках сокращает и расходы на покупку лицензионных программ и трудозатраты при создании коммерческого ПО. Не малую роль для достижения превосходного результата играет привлечение к разработке опытных специалистов. LVEE способствует появлению таких специалистов, развитию их навыков и расширению кругозора. Хотелось бы пожелать участникам конференции интересных проектов и максимум пользы от участия в LVEE.

Голос спонсора: Ciklum

Ciklum is a Danish innovative IT outsourcing company specializing in nearshore software development.

Established in 2002, Ciklum employs more than 1200+ IT specialists worldwide with more than 130+ current clients own development teams. Ciklum has pioneered a unique business model in Ukraine where employees have direct communication with the client and are equal to clients` home-based colleagues. In Ciklum we are working on the development of different kind of projects: from cross-platform end-user applications to multifunctional B2B scalable solutions.

What Ciklum offers to developers?

- Variety of knowledge sharing and training opportunities within Ciklum Knowledge Exchange community
- A lot of various technical seminars
- Unique working environment where you communicate and work directly with international businesses
- Possibility to work in a big and successful company
- Competitive salary
- Career and professional growth
- Long-term employment with 20 working-days paid vacation and other social benefits
- State of the art, cool, centrally located offices with warm atmosphere which creates really good working conditions

Ciklum has four development offices in the four largest cities in Ukraine & Belarus (Kyiv, Minsk, Kharkiv, Dnipropetrovsk, Donetsk) and two development offices in Pakistan (Lahore, Islamabad), as well as representative offices in Denmark, Sweden, United Kingdom, Switzerland, Germany and the Netherlands.

Join Ciklum and «Cross the Borders» together with us!

Голос спонсора: World of Tanks team

О проекте

World of Tanks (Мир танков) — первый MMO проект AAA класса, созданный белоруской командой. Разработчиками игра позиционируется как MMO-экшн с элементами ролевой игры, шутера и стратегии. Концепция «World of Tanks» базируется на массовых командных танковых сражениях в режиме PvP. Онлайн релиз русской версии игры состоялся 12 августа 2010 года, в марте 2011 года состоялся «китайский» релиз, а в апреле 2011 года проект успешно вышел на территории США и Европы.

Успех проекта World of Tanks можно оценить по целому ряду показателей: количество активных игроков более 2 миллионов, рекордная цифра одновременной игры — более 150.000 игроков на российском игровом кластере! В книгу рекордов Гиннеса мы вошли 23 января 2011 года с показателем 91 311 игроков он-лайн.

Дважды подряд в 2010 и 2011 году на Конференции Разработчиков Компьютерных Игр (Москва) наш проект был признан Лучшей клиентской он-лайн игрой (КРИ 2010) и Лучшей игрой (КРИ 2011).

В 2010 году по результатам международной выставки E3 (Лос-Анжелес) крупнейший MMO-портал Massivly назвал наш проект New Concept 2010. В настоящий момент определяются победители E3 2011. Мы сможем рассказать о наших успехах уже при личной встрече на конференции LVEE 2011.

Подробности о проекте

Игровые кластеры проекта находятся в дата-центрах России, Германии, США и Китая. Общее количество серверов в настоящий момент порядка 500, к концу года мы планируем удвоить их количество.

Как игровые, так и прочие инфраструктурные сервера функционируют на базе операционной системы CentOS.

Осенью 2010 года пики онлайна на российском игровом кластере достигали 30 тыс. В настоящий момент за счет высокотехнологичных решений наших специалистов (включающих в себя оптимизацию сетевой инфраструктуры, оптимизацию работы с базами

данных, оптимизацию нашего серверного ПО) пик он-лайна вырос до 150 тыс.

Высокая производительность достигается в т. ч. за счет использования free and open source software как технологической основы для работы серверной части самой масштабной MMO-игры, а именно: CentOS, MySQL, nginx, zabbix, nagios, cacti, python, django и т. д.

О нас

Над созданием проекта работает СООО «Гейм Стрим» — основной центр разработки компании `Wargaming.net`. История компании — это 12-летний опыт создания игр, более 15 выпущенных проектов, среди которых «Операция Багратион», а так же «Order of War», изданная Square Enix. В 2007 году мы объединились с минской студией Arise. В 2010 из разработчика превратились в издателя — мы сами осуществляем оперирование проекта World of Tanks.

Наши награды на Конференции Разработчиков Компьютерных Игр (Москва): лучшая стратегическая игра КРИ-2008 (проект «Операция Багратион»), приз от прессы КРИ-2009, лучшая компания-разработчик КРИ-2009 и КРИ-2010, приз от индустрии КРИ 2011, приз зрительских симпатий КРИ 2011.

Сейчас в студии работает более 200 человек. Мы готовимся к запуску в производство новых проектов и будем рады знакомству с талантливыми специалистами.

Нашим будущим сотрудникам мы предлагаем уникальную возможность работать над онлайн-проектами AAA класса, с применением передовых технологических решений; реализовать себя, работая над сложными задачами; учиться у ведущих специалистов отрасли. Со своей стороны мы создаём для этого все условия: комфортный офис, современное техническое обеспечение рабочих мест, все социальные гарантии, высокие белые зарплаты, обучение и т. д. А ещё у нас работают замечательные люди.

Наш контактный e-mail: `rabota@wargaming.net` Присоединяйтесь к команде World of Tanks!

Голос спонсора: инновационная компания Promwad

Инновационная компания Promwad разрабатывает электронику для массового производства: от аппаратного и программного решения до дизайна корпуса. Её клиенты — компании из России, стран Европейского союза, США и Канады. Роман Пахолков, основатель и руководитель инновационной компании Promwad, рассказывает об истории компании, полученном опыте и планах на будущее.

Promwad: история создания, численность и структура

Компания основана в 2004 году. Но в реальности ее история началась раньше, еще в 2002 году в процессе разработки новой универсальной измерительной платформы для серии приборов, используемых для измерений волоконно-оптических линиях связи. Вокруг этого амбициозного и сложного проекта собрались практически все белорусские специалисты по системам на кристалле (СнК) и операционной системе Embedded Linux.

На сегодняшний день в Promwad работает высококвалифицированная команда из 60 специалистов. В сентябре прошлого года мы открыли подразделение Promwad Mobile, нацелив его на разработку мобильных приложений для телефонов, планшетных ПК, ридеров, автонавигаторов и др. мультимедийной техники.

О бизнес-модели независимого дизайн-центра электроники

Я глубоко убежден, что такая бизнес-модель перспективна. Компания Promwad дорожит своей репутацией контрактного разработчика, предоставляющего услуги разработки на заказ и не претендующего на интеллектуальное владение продуктом или его частью. Во-первых, такой подход позволяет участвовать в проектах на острие технологий, обогащая знания наших инженеров, позволяя им впитывать мировой опыт. Во-вторых, изначально позиционируя себя как независимую компанию, мы построили эффективные бизнес-процессы, отобрали лучших специалистов и заставили такой бизнес быть рентабельным только за счет разработок. В-третьих, в условиях дефицита квалифицированных инженеров наши услуги при должном уровне сервиса постоянно растут в цене.

В компании Promwad всегда существовала четкая стратегия развития, следуя которой мы смогли выжить и стать полноценным развивающимся дизайн-центром. Ключевыми моментами этой стратегии являются следующие компоненты:

1. Нашей технической специализацией является проектирование аппаратуры на базе современных микропроцессоров и СнК, в основном с применением Embedded Linux, и стремление заниматься средними и крупными проектами в этих сферах.
2. Ориентация компании на привлечение лучших инженерных кадров и их постоянное развитие.
3. Наша клиентоориентированность и стремление к кооперации, не смотря на регулярные неудачи в этом процессе.

О работе с фрилансерами. Плюсы и минусы

Уже более 4-х лет мы не прибегаем к услугам внештатных специалистов, и все работы, связанные с разработкой электроники, выполняются штатными сотрудниками Promwad на полной занятости. Только такой подход позволяет получать гарантированные результаты по срокам исполнения и качеству проектов.

Своим начинающим коллегам, стремящимся к построению подобных компаний, я бы посоветовал никогда не надеяться на услуги фрилансеров в области основной деятельности, их работа не поддается измерению и не имеет гарантированного результата.

В то же время я хочу отметить, что при решении сложных инженерных, технических и алгоритмических задач мы прибегаем к использованию внешних консультантов, экспертов в узких областях. Но непосредственная реализация задач по проектированию выполняется штатными инженерами.

О планах развития компании Promwad

В заключении хотелось бы отметить, что мы разделяем разработку, производство и бренды на отдельные бизнесы. Так, компания Promwad всегда будет заниматься контрактной разработкой на острие технологий. Мы надеемся, что наши клиенты всегда будут довольны качеством наших услуг, получая свои дивиденды от продажи успешных продуктов, а мы сможем сохранить свою репутацию контрактного разработчика. Этот тот фундамент, который позволяет нам оценивать будущее электронной индустрии свежим взглядом.

Голос спонсора: компания Анакреон

Компания Anakreon основана в 2005 году и находится в Минске, Беларусь. В настоящее время штат компании насчитывает более 40 сотрудников..

Компания Анакреон оказывает услуги по разработке, тестированию и поддержке программного обеспечения. Направления деятельности можно условно разбить на:

- **Интернет приложения**
- **Приложения для управления данными**
- **Моделирование и разработка пользовательских интерфейсов**
- **Тестирование программ**
- **Хостинг и поддержка программ**

В течение всего времени существования, наша компания последовательно нацелена на использование систем с открытым кодом (OpenSource). На настоящее время мы приобрели огромный опыт работы с такими системами и утверждаем, что OpenSource с успехом может применяться для поддержки бизнес процессов компаний различного калибра.

Нас можно найти по адресу:

ООО "Анакреон"
Республика Беларусь
г.Минск, ул.Сухая д.4 офис 3а
тел/факс: +375 17 203 3055
e-mail: info@anakreon.by
internet: www.anakreon.by

Интервью с участниками

По сложившейся традиции в сборник материалов включены интервью, взятые представителями оргкомитета у участников конференции: как сразу после LVEE 2010, так и перед началом LVEE 2011. Участники рассказывают о себе и своей работе, делятся планами, высказывают мнения по актуальным вопросам, волнующим сообщество.

1 Николай Маржан, Киев, Украина (LVEE 2010)

Николай Маржан: Я работаю в компании PortaOne. Вот уже 3.5 года. Мои обязанности — это релиз-инжиниринг и, на данный момент, обеспечение работы системы мониторинга, которую я разработал с нуля.

LVEE: А чем занимается компания?

H: Мы разрабатываем свои VOIP-решения, основной продукт это VOIP-биллинг и, соответственно, все сопутствующие сервисы.

L: В работе вам приходится использовать свободное ПО?

H: Конкретно в моей работе — да. Я это очень приветствую, и все свои наработки мы отдаем под той же лицензией (GPL) нашим клиентам. Т. е. если клиент купил наше решение, мы отдаем исходные коды, а если там GPL — под ней же свои доработки.

L: А как вообще возникла твоя симпатия к свободному ПО?

H: Трудно сказать. Проще вспомнить, как появилась симпатия к Unix-way. В 16 лет я написал свой собственный форум на perl. И после этого влюбился в Unix. На бесплатном хостинге `agava.ru` тогда были только PHP и perl. Я посмотрел на PHP и понял, что это не мое. А perl... Не считаю себя в нем гуру, но он мне очень нравится.

L: И так мир Unix просочился...

H: Через эту лазейку.

L: Сейчас тебе приходится вплотную заниматься системным администрированием?

H: Как раз системное администрирование от меня ушло почти полностью, кроме отладки системы мониторинга, и 95% времени я яв-

ляюсь разработчиком. Занимаюсь релиз-инжинирингом, т.е. подготавливаю наш продукт к деплойменту. Готовлю инсталляционные диски, апгрейд-процедуры, в общем делаю все, чтоб деплоймент прошел как можно легче.

L: А мониторинг?

H: На прошлой работе я 50% времени занимался разработкой системы мониторинга, и вот, на новой — решили использовать накопленный опыт и предложили написать ее полностью с нуля.

В маленьких компаниях мы идем от готовых решений к реализации. В PortaOne я смог отталкиваться от того, что нужно потребителю — в данном случае отделу технической поддержки — к тому, как это реализовать. Невзирая на сроки, ориентируясь исключительно на качество. Это была главная идея, и благодаря ей я смог реализовать все что было задумано, без недостатков моих ранних систем мониторинга.

L: Какое свободное ПО было задействовано в процессе?

H: Из свободного ПО был использован проект nagios, был использован транспорт nagios'а, а это отдельный проект. Были позаимствованы кое-какие скрипты построения графиков. Но на данный момент nagios можно заменить буквально тремя тысячами строк кода на perl, он сейчас используется просто как планировщик заданий. Т. е. разработка переросла стадию инфраструктуры, связывающей несколько проектов, и стала самостоятельным проектом.

L: За пределами работы ты, вероятно, тоже пользуешься свободным ПО?

H: Да, на своем десктопе 3 года использовал FreeBSD, но потом неудачно купил ноутбук :). Так что пришлось поставить на него Ubuntu. И теперь она на всех моих машинах.

L: Переход с FreBSD дался легко?

H: Настолько легко, что трудно даже представить (это ведь не наоборот). На самом деле я считаю себя скорее BSD-гуру, чем Linux. Я мэйнтейнер нескольких портов FreeBSD и посто ее люблю.

О портах... На самом деле они исходят из моих потребностей по работе. Что-то было нужно — сделал порт. А потом подготовил его для включения в мэйнстрим. Мы, как комьюнити, должны отдавть свою работу — иначе... Иначе мы не достигнем вселенского счастья. Если я отдам свою крупицу сообществу, кто-то ей воспользуется.

2 Наим Шафиев, Москва, Россия (LVEE 2010)

Наим Шафиев: Я из Москвы, работаю в компании ECUmoney Limited, параллельно учусь. Я могу себя позиционировать как поборник свободного ПО в его истинном проявлении, полностью без несвободных лицензий. Интересуюсь perl, пишу на нем, иногда приходится работать и с PHP, и с Pithon, и, боже мой, даже с Delphi.

Но на самом деле я придерживаюсь главной теории экономики, что разделение труда — это великое благо, потому что при разделении труда выделяются наиболее способные люди для определенных вещей, соответственно повышается общая производительность труда. Соответственно, мой основной язык программирования это perl, я на нем программирую с 2001 года, но плотно занялся года три назад, можно сказать, к нему возвратился. Почему это так? Надо сказать, что perl 2001 года был не очень хорош, особенно для web. Использование CGI-обработки приложений является не очень хорошим способом обработки. Оно слишком низкоуровневое. Просто на тот момент ничего больше не было. Тот же PHP был посто ужасен на самом деле — он ничего не поддерживал. Кстати, я доклад прочитал миенно по поводу модернистического perl [на LVEE 2010], там я указал некоторые базовые направления, по которым стоит развиваться и на что стоит смотреть.

L: А свободное ПО — это больше профессиональный интерес или личные предпочтения?

Н: Я пытаюсь пользоваться только открытыми решениями, начиная от видеокарт и заканчивая... буквально всем. Т. е. если под устройство отсутствуют свободные драйвера — я его не использую. По крайней мере, стараюсь: сейчас все-таки приходится использовать Radeon, там над открытыми драйверами, насколько я знаю, сейчас работают всего два челоаека... Что в принципе ужасно.

О профессиональной деятельности. В ECUmoney Limited мы используем практически полностью открытый стек. Используется CentOS. Хотя у нас был сервер на RHEL, но я считаю, что нет смысла платить за поддержку, которую я могу выполнять сам. ECUmoney Limited — мое основное место работы — это проект, который занимается веб-деньгами. Это новозеландский банк, и он осуществляет все операции вплоть до депонирования и работы на бирже. Менеджмент в Москве, соответственно там же я и работаю.

L: Эта строгая позиция в отношении свободного ПО и непримиримость к двойным стандартам — как она возникла?

H: С детства интересовался экономикой, особенно политэкономией. В принципе очевидно, что любые патенты, ограничения, проприетарщина — это ограничения свободы, и за счет этого происходит ограничение развития человечества. Могу привести одну цитату. Билл Гейтс, человек, который до недавнего времени руководил самой большой компанией по разработке проприетарного ПО, говорил: «Если бы патенты существовали в том виде, в каком они существуют сейчас — никакая индустриальная революция, которая полностью перевернула жизнь людей в Европе в XVII веке, не прошла бы».

Что касается Unix-way — в 2001 году я начал изучать computer science, прочитал одну американскую книжку про альтернативные операционные системы. Тогда найти диск с FreeBSD даже в Москве было довольно сложно, но я нашел и попытался его установить. С первого раза конечно же ничего не получилось. И жесткий диск я стер (но это ничего, я отношусь к тем админам, которые все-таки делают бэкапы). Пришлось распечатать весь хэндбук, он был действительно огромный... И вот так я потихонечку пересел на свободное ПО. Сначала это была FreeBSD, потом Mandrake, потом Fedora, была еще пара тестовых дистрибутивов. Писал даже свой дистрибутив для встраиваемых решений. А в конце остановился на Debian: решил, что раз они наиболее сильные поборники свободного ПО, стоит на нем остановиться.

L: А участие в свободных проектах?

H: В perl есть CPAN — репозиторий свободных проектов. Я в CPAN-е, можете набрать `search.cpan.org/~naim/` и увидите мой проект. На самом деле у меня разработок довольно много, но часть пока еще не готова для релиза. Моя разработка AnyEvent Benchmark — это довольно удобный фреймворк для осуществления собственных HTTP и HTTPS-тестинга, для тестирования SOAP и других вещей. Сейчас она в разработке, я сам ее использую, как и еще пара членов `moscow.pm.org/`.

L: А как ты оказался на LVEE?

H: Это довольно сложно. Информация о LVEE несколько раз появлялась на `linux.org.ru`, а я на этом ресурсе с 2003 года. Первый раз хотел приехать еще в 2009 году... Для меня оказалось просто шоком, что в Беларуси проводится настолько прогрессивное мero-

приятие, формат которого на мой взгляд может в чем-то соперничать с FOSSCON, плюс уровень разработчиков довольно хороший.

3 Wolfgang Spraul, Гонконг, для LVEE

В рамках LVEE 2010 было также взято онлайн-интервью у генерального директора гонконгской компании-стартапа ООО Sharism, известной своими разработками в области свободного аппаратного обеспечения (САО). Вопросы от имени LVEE, кроме отдельно оговоренных случаев, задавал Максим Мельников.

Wolfgang Spraul: Пожалуйста, обратите внимание: я считаю, что это интервью действительно очень важно для нас и САО. СПО-движение в России и странах бывшего СССР, таких как Беларусь, существует давно и является очень активным. К сожалению, таможенные пошлины в этих странах имеют достаточно жесткую историю, поэтому русским фанатам СПО тяжело принимать участие в САО-движении. Но, я верю, что мы придем и сюда.

В общем:

W: Я, Wolfgang Spraul — борец за свободу :-) А если серьёзно, я инженер, который почувствовал, что разрабатывать аппаратуру на тех же принципах, что и СПО, можно намного более успешно, чем традиционным проприетарным способом. Мне 35 лет, я живу в Китае и пытаюсь сделать САО возможным, как в техническом плане, так и в плане бизнеса. И еще я питаю пристрастие к Starbucks.

L: Что есть «Sharism»? Что есть «Qi-hardware»? Что есть «Ben Nanonote»?

W: Qi-hardware — это свободное аппаратное обеспечение (единственное объединение проектов «свободного железа», которое я знаю). А свободное «железо», это «железо», которое сделано на основе свободных знаний, с использованием тех же принципов, что и СПО. Некоторые элементы, такие как чипы памяти NAND и ПЗУ, ЖКИ — не будут свободны ещё долго. Но, как минимум, мы начали работать над полностью свободным процессором с Milkymist project.

ООО Sharism — маленькая Гонконгская компания-стартап с 5 сотрудниками. Мы производим свободное оборудование в Китае, а это означает, что мы делаем оборудование без вендорских заложенных модулей. Все наши знания и вся информация доступна, мы используем только легко заменяемые компоненты, только ПО под

GPL-лицензиями и т. д. Мы работаем как OEM / ODM, так что теоретически можем создать любой продукт, но практически имеет смысл работать на стандартной технологической базе, в нашем случае — на основе процессоров Ignetic XBurst.

Мы можем делать GPS-приемники, цифровые фоторамки, цифровые камеры, телефоны, даже датчики напряжения и влажности.

Ben Nanonote — переносной карманный компьютер, похожий на маленький электронный словарь. Вес 126 г. с батареей. Он на 100% работает под управлением СПО. Сейчас на ядре Linux 2.6.32. Мы отправили все патчи в upstream ядра 2.6.34, так что, возможно, через несколько месяцев ядро с kernel.org будет на 100% поддерживать наше устройство. Характеристики — процессор 366МГц, MIPS, 32 Мб ОЗУ, экран 3", отсутствуют usb-хост, wifi, bluetooth.

L: Как вы решили организовать вашу компанию? Правы ли те, кто называют вас «fork-ом» OpenMoko?

W: Компания ООО Sharism появилась в ноябре 2009-го, когда стало ясно, что другие маленькие компании будут заинтересованы присоединиться к проекту Qi-CAO, и нам захотелось иметь отдельную маленькую компанию как азиатского производителя. Некоторые из нас в прошлом работали для OpenMoko, но CAO — это новая идея, и я не думаю, что OpenMoko когда-либо будет участвовать в CAO.

Разработка свободной аппаратуры:

L: Какая самая большая проблема в разработке открытой аппаратуры?

W: Поиск клиентов, которые понимают долгосрочную перспективу, покупают наше простое маленькое устройство сегодня (такое, как Ben Nanonote) и помогают нам в разработке СПО для этого устройства. На данный момент мы продали 800 Ben Nanonote.

Как открытое «железо» может приносить деньги? Так же, как и проприетарное — через продажу высококачественных устройств, создание сильного бренда, лояльность клиентов благодаря хорошему сервису и т. д.

L: Можешь привести простые расчёты, как «железная» компания может быть успешной? Цена устройства, доход, число проданных экземпляров, размер компании?

Это очень сложно, потому что есть разовые расходы, и есть расходы на каждую единицу. Также очень важен потенциал зарабатывания денег на сервисах, мелочах, и т. д.

Мы потратили около $180 000 наличными, чтобы сделать первую 1 000 Ben Nanote. Итак, если мы продадим 1 000 за $99, мы всё равно в минусе. Возможно, если получится продать 2 000 или 3 000, это будет выглядеть лучше.

L: СПО даёт возможность объединять разработки — новые устройства получают большую базу ПО с самого начала. А что открытость «железа» даёт «железному» стартапу?

То же самое. Сейчас в мире «железа» многие вещи изобретаются снова и снова. Объединив усилия, всего можно добиться намного лучше, это позволит производить лучшие продукты за более выгодную цену и быстрее выходить на рынок. Огромная неэффективность «железного» мира может быть преодолена благодаря использованию философии нашей компании, Sharism :-)

OOO Sharism:

L: Как много людей работает в компании? Кто вы в плане структуры компании и организации?

W: 5 работников на полный рабочий день. В такой маленькой команде мы все помогаем друг другу, структура размыта, мы все привносим в команду наш уникальный опыт.

L: Какую последнюю цель вы успешно достигли? Какая следующая?

W: Последняя достигнутая — продать 400 Ben Nanonote к концу марта, была успешно выполнена. Следующая — продать 1 000 Ben Nanonote к концу июля.

L: ПО очень важно. Как вы разрабатываете ПО?

W: В большинстве случаев мы полагаемся на сообщество СПО, у нас нет денег для оплаты разработки ПО. Сообщество OpenWRT было особенно полезным, теперь мы видим некоторый интерес со стороны Jlime и некоторых Debian-разработчиков.

Daniel A. Nagy, ePoint System: Устройства какого типа (телефоны, планшеты, нетбуки?) вы планируете произвести в будущем? У вас есть прогнозный график выпуска?

W: Мы можем произвести что угодно на базе процессоров Ingenic XBurst. Мы изучаем расширенные NanoNote, GPS-устройства, цифровые фоторамки, телефоны, программируемые FPGA-матрицы, а также датчики электричества и воды. Если вы хотите что-нибудь в этом роде, с минимальным количеством в 5 000 единиц или около того, мы готовы сделать это в стиле CAO.

D: Насколько совместимыми друг с другом вы планируете делать свои устройства, с точки зрения как аппаратного, так и программного обеспечения?
W: Унификация очень важна. Чтобы оценить ее, мы сначала смотрим на наиболее сложные части, особенно в отношении программного обеспечения. Например, мы учитываем ядро Linux, дистрибутив OpenWrt и т. д. Мы рассматриваем совместимость с точки зрения ПО. С точки зрения аппаратуры наиболее сложные части — процессор Ingenic XBurst и FPGA от Xilinx.
L: Получали вы комментарии от больших компаний?
W: Ничего серьёзного.

Аппаратура:

D: Вопрос о модульности: будет ли неприемлемо дорогим с точки зрения денег, времени жизни батареи или веса, сделать все части пригодными к самостоятельному использованию: экран, клавиатуру, тач-скрин, модуль 3G и т.д., так чтобы они соединялись с материнской платой через USB (не обязательно через стандартные разъемы, хотя бы на уровне сигналов)?
W: Модульность не работает. Аппаратная индустрия вкладывает значительные однократные инвестиции, чтобы добиться интеграции и низкой стоимости устройств. Теоретически все можно интегрировать на одной кремниевой пластине. Цена производства кремниевой пластины может быть всего около 50 центов, если не смотреть на огромные однократные вложения, необходимые чтобы достигнуть этого уровня интеграции.

Ben Nanonote:

L: Энергопотребление. Как долго он может работать?
W: Мы получаем отчёты от пользователей, говорящие о 6-15 часах, в зависимости от нагрузки.
Richard_Ferlow, `habrahabr.ru`: Что пользователь может делать со своим Nanonote?
W: Наш последний OpenWrt-образ от 7 мая 2010. Что работает сейчас — это консольные приложения такие как vi или mutt, или музыкальный плеер GMU, сделанный на SDL (он может проигрывать

только Ogg Vorbis, а не патентованный MP3). Чтобы получить список OpenWRT-приложений и их статус в Ben Nanonote, можно посмотреть «Приложения» на сайте. Также вы можете запускать дистрибутив Debian или JLiMe, который с недавних пор поддерживает Ben Nanonote: `http://jlime.com/mw4/index.php/Faq_nanonote`.

Так или иначе, потребуется время, чтобы сделать приложения, которые действительно будут хорошо работать на устройстве, а не просто их собрать.

habrahabr.ru: Почему экран намного меньше верхней части Ben NN? Будет ли следующий Nanonote иметь более высокое разрешение?

W: Экран стандартный, 3" TFT, обычно этот размер встречается в цифровых камерах. Другой популярный размер — 3.5", но он не вмещается в габариты устройства. Экраны 3.2" редки и дороги, а цветные экраны специального размера совершенно недоступны для маленьких компаний, таких как мы. Когда мы ищем инновации, мы в первую очередь смотрим на СПО, свободные технологии. Что касается TFT в Ben Nanonote, это Ilitek ILI8960 drive IC, с открытой спецификацией.

Когда мы думаем над улучшением экрана, мы пытаемся найти лучший экран с таким же IC, или улучшенным (следующей версии) Ilitek drive IC.

Таким образом мы сможем быстрее использовать стабильную апстрим-версию Linux-драйвера. Если у нас будет возможность выбирать между новым экраном: один большой, полупрозрачный и т. д., но с закрытыми спецификациями и проприетарным драйвером, или другой более простой, размером 2.8 дюйма, но с открытой спецификацией — мы выберем более простой ЖКИ-экран. Мы изобретаем свободу, мы хотим получать радость от технологий и изменять их для наших клиентов без спросов и разрешений.

4 Алексей Кондратенко, Минск, Беларусь (LVEE 2011)

За несколько дней до конференции организаторы пообщались с одним из докладчиков LVEE-2011, Алексеем Кондратенко. Алексей — активный контрибьютор в open source-проекты, в частности Membase (Couchbase), ведущий разработчик Альторос Девелопмент. Во время беседы он поделился мыслями о выборе лицензий, тенден-

циях open source-проектов, недавней поездке в Кремниевую долину и дал несколько советов начинающим Linux-разработчикам.

LVEE: Алексей, расскажи немного о себе.

Алексей: Программирую. Много.

L: В каких свободных проектах ты принимал участие?

A: Мне довелось портировать один полупроприетарный драйвер программного модема в Linux 2.6. Я поучаствовал немного в разработке Ruby on Rails, а также написал замечательный навигатор по проекту и документации (gpicker). Но, разумеется, самое значительное, — это мое участие в проекте Membase. И особенно приятно, что это работа за деньги, но над free software.

L: Какой из этих проектов тебе наиболее интересен с профессиональной точки зрения?

A: В каждом проекте или проектике есть свои изюминки. Я считаю, что не бывает неинтересных задач. Бывают только неинтереснее решения. Я считаю, что очень важно научиться получать удовольствие от качественно сделанной работы. Тогда любая, даже самая тривиальная задача, будет приносить удовольствие. В компании, где я работаю, созданы именно такие условия для самореализации.

L: Что ты думаешь об основных лицензиях на распространение ПО? Сдерживают ли по-твоему лицензионные ограничения развитие open source-продуктов?

A: Два больших и очень различных класса — copyleft и non-copyleft.

Первый класс — лицензии вроде GPL, LGPL и AGPL. Эти лицензии обеспечивают публичность модификаций. GPL, например, требует, чтобы параллельно с предоставлением готовой программы (оригинальной или модифицированной) предоставлялся доступ к исходным кодам. Это не позволяет корпорациям развивать свои закрытые форки. К сожалению, я был свидетелем того, как люди, на мой взгляд неоправданно, опасаются этого класса лицензий. Я слышал, будто бы GPL-код помешал покупке нескольких стартапов. Всем известно, что Android практически целиком избегает GPL (кроме неизбежного GPL для ядра).

Второй класс не накладывает copyleft-ограничений. Он позволяет компаниям «наживаться» на закрытых версиях open source-проектов. Хотя обычно скрывать код не очень выгодно. Так что сторонники этого класса как правило упирают на большую либеральность, саму возможность делать с кодом все что угодно.

Кроме copyleft vs. non-copyleft есть еще много всяких тонкостей, например, борьба с DRM и патентная волокита.

Выбор или смена лицензии может быть поводом для создания альтернативного проекта с более мягкой лицензией. Например, компания Apple полным ходом идет к отказу от GCC в пользу основанного на LLVM набора компиляторов Clang. Сейчас они поставляют очень старые версии GCC, GDB и других частей GNU Toolchain. Одна из причин — переход данных проектов на GPL3. Есть основания полагать, что Apple по той же причине готовит замену Samba.

L: Поделись своими мыслями, за счёт чего идёт развитие open source-проектов? В смысле финансовой стороны.

A: Есть разные модели бизнеса на free software. В последнее время большее распространение получает модель, при которой клиенты покупают не сам софт, а подписку на поддержку. RedHat довольно успешно зарабатывает миллиарды в год на этой модели.

Но голый энтузиазм никогда не пропадет полностью. На github, например, можно найти кучу очень впечатляющих проектов, которые люди делают в свое личное время.

L: При сравнении open source-проектов с коммерческими, в каком случае качество проигрывает и почему?

A: Я считаю, что это независимые вещи. Можно одинаково хорошо или плохо делать и свободный и проприетарный софт. Свободный софт дает мне ряд возможностей: видеть, что внутри, исправлять это, и наконец, использовать его куски в других проектах.

Но качество всегда определяется личностными характеристиками и упорством инженеров, которые над кодом работают.

L: Всегда ли open source-проекты бесплатны?

A: И да, и нет. Всегда можно просто взять и применить любой свободный софт. Но внедрение любых технологий в организациях требует некоторых затрат на обучение/поддержку и так далее. И этого невозможно избежать.

L: Какое, на твой взгляд, будущее ждет open source?

A: Теперь уже очевидно, что open source — это просто часть мейнстрима. Будущее таких важных инициатив как GNU Linux (особенно на десктопах) по-прежнему не очевидно. Но сам факт повседневного применения свободного софта уже является неотъемлемой частью нашего программистского дела. Да, в принципе, и повседневного быта обычных людей.

Также очень ценна доступность кода фреймворка или библиотеки, которую используешь. Похоже, что не только я один привык иметь возможность глубоко «копать», разбираясь с проблемой. Это связано с тем, что часто баги встречаются на стыке технологий.

Сейчас все меньше остается проприетарных и закрытых технологий (а интересных так и вообще нет). Я думаю, что люди просто все больше привыкают полагаться на наличие кода. И совсем скоро это будет просто must have.

L: Чем ты пользуешься в работе?

A: Я большой сторонник GNU/Linux ифилософии free software в частности. Для работы использую Emacs и кучу великолепных маленьких утилит и утилиток, которыми так славен free software world.

L: Книга которую должен прочесть каждый программист?

A: Сложно сказать. Есть множество очень классных книжек, которые можно рекомендовать по разным причинам. Думаю «классический» выбор в пользу TAOCP и SICP никого не удивит.

L: А без какой художественной книги ты бы не стал программистом?

A: Я стал программистом не из-за художественной литературы. Когда я был в третьем классе, отец собрал Baltic, советский клон культового ZX Spectrum. Играть на нем, разумеется, было очень интересно. Потом у меня начали получаться простые программы, из которых постепенно и вырос интерес к программированию.

L: Какой профессиональный совет ты можешь дать начинающим программистам?

A: Я вижу, как люди сталкиваясь с проблемами, скажем, в Rails или в каком-либо плагине, не проявляют должной активности для исправления их в upstream. Иногда разработчики отправляют патчи, которые потом просто пропадают. Очень важно понять, что правильный и завершенный патч надо еще правильно оформить и подать людям, которые отвечают за проект. И это также очень важная и непростая работа. Главное — не бояться и помнить, что в любом проекте есть правила подачи патчей. Если все делать правильно и не сдаваться, эффект будет не только для самолюбия, но и для всего человечества, как бы пафосно это ни звучало.

L: Алексей, ты только что вернулся из Кремниевой долины в Калифорнии, самой Мекки программистов. Как впечатления? Расскажи, где тебе работалось лучше.

A: Непосредственно работа там получается лучше. Наверное, это обусловлено тем, что полностью отсутствуют отвлекающие факторы. А в целом, большой разницы с работой, например, в офисе Altoros не чувствовал. Но здесь мне однозначно нравиться больше.

Научное издание

ОТКРЫТЫЕ ТЕХНОЛОГИИ

Сборник материалов
седьмой международной конференции
разработчиков и пользователей
свободного программного обеспечения
Linux Vacation / Eastern Europe 2011

(Гродно, 30 июня –03 июля 2011 г.)

Ответственный за выпуск Д.А. Костюк
Технический редактор Н.Е. Фицнер
Фото на обложке А. Гончаренко

Подписано в печать 21.06.2011.
Формат 60×84 $^{1}/_{16}$. Бумага офсетная.
Ризография. Усл. печ. л. 7,7. Уч.-изд. л. 5,5.
Тираж 215 экз. Заказ 3084.

Издатель и полиграфическое исполнение:
частное производственно-торговое унитарное предприятие
«Издательство “Альтернатива”».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий
ЛИ N 02330/0494051 от 03.02.2009.
ЛИ N 02330/0150460 от 25.02.2009.
Пр. Машерова, 75/1, к. 312, 224013, Брест.