Наукове видання

**ПРОГРАМНА АРХІТЕКТУРА КОМП'ЮТЕРА:
АКТУАЛЬНІ ЗАДАЧІ ТА НАУКОЄМНІ ТЕХНОЛОГІЇ**
Збірник тез доповідей науково-практичного семінару
31 жовтня 2013 року, м. Кіровоград

Відповідальний редактор Дóренський О.П.

Формат 60x84/16. Гарнітура Times New Roman. Ум. друк. арк. 2,0. Зам. № 741

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КІРОВОГРАДСЬКИЙ НАЦІОНАЛЬНИЙ
ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кафедра
програмного забезпечення

Студентське
наукове товариство

Науково-практичний семінар

"Програмна архітектура комп'ютера: актуальні задачі та наукоємні технології"

31 жовтня 2013 року

Збірник тез доповідей

Кіровоград
КНТУ
2013

УДК 004
ББК 32.97
П91

П91 Програмна архітектура комп'ютера: актуальні задачі та наукоємні технології : Збірник тез доповідей науково-практичного семінару, м. Кіровоград, 31 жовтня 2013 року. — Кіровоград: РВЛ КНТУ, 2013. — 34 с.

Схвалено та рекомендовано до друку кафедрою програмного забезпечення
Кіровоградського національного технічного університету
/ протокол від 6 листопада 2013 року № 7 /

Збірник містить матеріали науково-практичного семінару "Програмна архітектура комп'ютера: актуальні задачі та наукоємні технології", який відбувся 31 жовтня 2013 року у Кіровоградському національному технічному університеті.

УДК 004
ББК 32.97
П91

Відповідальний за випуск: Дóренський О.П.

Матеріали збірника публікуються у авторській редакції.
Відповідальність за зміст несуть автори.

Адреса оргкомітету
Кіровоградський національний технічний університет
Кафедра програмного забезпечення
просп. Університетський, 8, м. Кіровоград, 25006
Тел.: (0522) 39-04-49. E-mail: conf_kir@ukr.net

ЗМІСТ

УДК 004.75

Н.В. Підгорна
Науковий керівник — Мелешко Є.В., канд. техн. наук, доцент
Кіровоградський національний технічний університет

Архітектура та застосування Grid-систем

Останнім часом до обчислювальних кластерів проявляється підвищений інтерес з боку науки, освіти та промисловості. Кластерні обчислювальні системи стають повсякденним інструментом дослідника та інженера. У зв'язку із збільшенням кількості кластерів набирає все більшої популярності концепція Grid-систем [1]. Grid підтримують спільне і скоординоване використання різноманітних ресурсів в динамічних, розподілених віртуальних організаціях, дозволяючи з географічно розосереджених компонентів, що застосовуються в різних організаціях з різними правилами роботи створювати віртуальні обчислювальні системи, здатні спільно підтримувати необхідний рівень обслуговування. У Grid можуть об'єднуватися різні обчислювальні ресурси: персональні комп'ютери, робочі станції, кластери і супер-комп'ютери.

Метою роботи є дослідження архітектури GRID-систем та сфери їх застосування.

Для того щоб розглянути архітектуру GRID-систем доцільно зробити декомпозицію системи в двох «площинах»:

- вертикальна архітектура;
- горизонтальна архітектура.

Рисунок 1 – Вертикальна архітектура GRID-системи

З точки зору вертикальної архітектури, GRID являє собою стек протоколів, заснованих на Інтернеті (рис. 1) [2], тому необхідною умовою приєднання ресурсу є підтримка цих протоколів.

Рівень Фабрикатів надає ресурси, спільний доступ до яких забезпечується через протоколи GRID [2].

Рівень Зв'язку визначає базові комунікаційні та ідентифікаційні протоколи, необхідні для проведення специфічних для GRID операцій (транзакцій). Комунікаційні протоколи дозволяють здійснювати обмін даними між ресурсами рівня Фабрикатів. Ідентифікаційні протоколи, побудовані на

комунікаційних сервісах, надають криптографічно захищений механізм для ідентифікації користувачів і ресурсів [2].

Рівень Ресурсів базується на комунікаційному і авторизаційному протоколах рівня Зв'язку. Він визначає протоколи для:

- здійснення безпечного обміну інформацією;
- ініціалізації, моніторингу та проведення спільних операцій на індивідуальних ресурсах;
- створення облікових записів користувачів;
- проведення обліку утилізованого часу для кожного з користувачів [2].

На рівні Кооперації згруповані протоколи і сервіси, які не пов'язані з будь-яким конкретним ресурсом, є більш глобальними за природою і забезпечують колективну взаємодію ресурсів [2].

Рівень Додатків включає в себе додатки користувача, які функціонують в середовищі GRID [2].

Під горизонтальною архітектурою будемо розуміти сукупність сервісів, розміщених на рівні додатків (Application) (рис. 2).

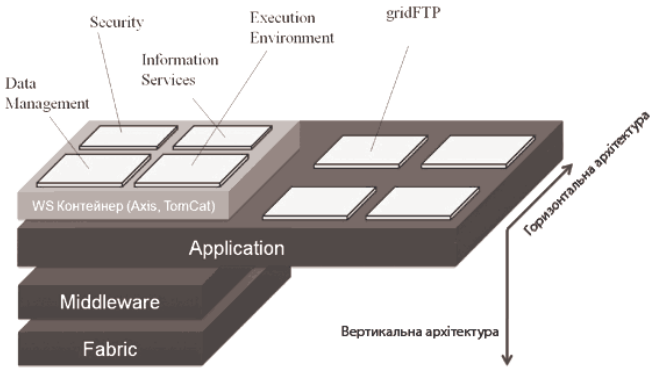

Рисунок 2 – Вертикальна і горизонтальна архітектура GRID

Для організації взаємодії в рамках GRID-системи на цьому рівні реалізовані сервіси віддаленого доступу: запуску завдань, пересилки файлів, моніторингу завдань, авторизації та аутентифікації і т.д. У міру розвитку мережі Інтернет з появою технології веб-сервісів [4] для досягнення універсальності та підвищення інтероперабельності компонентів більшість функцій в GRID-системі реалізовано у вигляді сервісів (рис. 2). Основним стандартом, що описує сервіс-орієнтовану архітектуру GRID-системи, є Open Grid Services Architecture (OGSA) [3], яка визначає основні служби, що функціонують на рівні додатків і необхідні для побудови GRID:

- Execution Management services (виконавча підсистема);
- Data services (служби для роботи з даними);
- Resource Management services (служби управління ресурсами);
- Security services (служби забезпечення безпеки);
- Information services (служби моніторингу та інформації);
- Self-management services (служби самоврядування).

Найчастіше остання група служб (Self-management) інтегрується в перші п'ять і в явному вигляді стандарт не вимагає її окремої реалізації в GRID.

Таким чином, будь-яка обчислювальна одиниця або ресурс (кластер, окремий комп'ютер, сервер баз даних і т.д.), що підключається до GRID, повинен:

- підтримувати вертикальну і горизонтальну архітектуру GRID;
- належати до однієї або більше груп служб (стандарт не обумовлює і не встановлює обмеження на кількість служб, встановлених на одному комп'ютері, єдине обмеження - це зниження продуктивності);
- зареєструватися у віртуальних організаціях, в рамках яких він функціонуватиме (вузол може функціонувати в рамках декількох віртуальних організацій, найчастіше використовується саме такий підхід).

Найбільш характерними властивостями цього інформаційно-обчислювального середовища є [5]:

- масштаби обчислювального ресурсу (обсяг пам'яті, кількість процесорів), які багаторазово перевершують ресурси окремого комп'ютера або одного обчислювального комплексу;
- гетерогенність середовища, до її складу можуть входити комп'ютери різної потужності, що працюють під управлінням різних операційних систем і зібрані на різній елементній базі;
- просторовий (географічний) розподіл інформаційно-обчислювального ресурсу;
- об'єднання ресурсів, які не можуть управлятися централізовано (у разі, якщо вони не належать одній організації);
- використання стандартних, відкритих, загальнодоступних протоколів і інтерфейсів;
- забезпечення інформаційної безпеки.

За своїм призначенням GRID прийнято ділити на обчислювальні системи (computational GRID) і системи, орієнтовані на зберігання великих масивів інформації (data GRID).

До прикладних задачах, які можуть використовувати GRID, зокрема, відносяться:

- складне моделювання;
- спільна візуалізація дуже великих наборів наукових даних;
- розподілена обробка в цілях аналізу даних;
- зв'язування наукового інструментарію з віддаленими комп'ютерами та архівами даних.

Мабуть, найбільш ефективним є застосування GRID для вирішення наступних завдань:

- розподілені високопродуктивні обчислення, рішення дуже великих завдань, що вимагають максимальних процесорних ресурсів, пам'яті і т.д.;
- «високопоточні» обчислення, що дозволяють організувати ефективне використання ресурсів для невеликих завдань, утилізуючи комп'ютерні ресурси, що тимчасово простоюють;
- проведення великих разових розрахунків;
- обчислення з залученням великих обсягів розподілених даних, наприклад, в метеорології, астрономії, фізиці високих енергій;
- колективні обчислення – одночасна робота декількох взаємодіючих

завдань різних користувачів.

Аналіз світового досвіду побудови GRID-систем показує, що вони лежать в основі рішення наступних проблем:

- об'єднання різнорідних систем;
- спільне використання даних;
- динамічне виділення ресурсів;
- переносимість додатків в гетерогенному середовищі;
- забезпечення інформаційної безпеки.

Висновки. Розглянуто архітектуру GRID-систем, їх характерні властивості та задачі, для рішення яких можна використовувати ці системи. На основі проаналізованих даних можна сказати, що GRID - це якісний розвиток системи розподілених обчислень, яка заснована на найбільш доцільному використанні ресурсів. У GRID користувач отримує доступ до ресурсів як спеціальний електронний сертифікат, а ця "розумна" система сама регулює пошук вільних ресурсів, звернення до сховищ даних в рамках своєї віртуальної організації.

Список літератури

1. Фостер І., Кесельман К., Нік Дж., Тьюке С. The physiology of the grid an open grid services architecture for distributed systems integration. – 2003.
2. Фостер І., Кесельман К., Тьюке С. Анатомія Grid: Включення масштабованих віртуальних організації / / Міжнародний журнал високопродуктивних обчислювальних систем, 15 (3). - 2001- С. 200-222.
3. The Open Grid Services Architecture / [Фостер І., Кісімото Х., Сава А. та ін..] - версія 1.5 - 24 липень 2006 року.
4. http://en.wikipedia.org/wiki/Web_service
5. Чайковський К., Фіцджеральд С., Фостер І., Кесельман К. GRID Information Services for Distributed Resource Sharing. – 2001.

УДК 004.451.9

А.В. Підгорна
старший викладач
Кіровоградський інститут державного та муніципального управління
Класичного приватного університету

Принципи побудови платформи Android та перспективи її подальшого розвитку

Мобільні пристрої стали невід'ємною частиною повсякденного життя і діяльності більшості людей в усьому світі. Тому операційні системи для мобільних пристроїв в даний час бурхливо розвиваються. До мобільних пристроїв прийнято відносити мобільні телефони, смартфони, планшети і комунікатори. Розробники ОС для мобільних пристроїв працюють над тим, щоб наблизити можливості цих ОС до можливостей ОС для настільних і

портативних комп'ютерів. Проте в ОС для мобільних пристроїв є своя специфіка. Їх основні особливості наступні:

- облік більш жорстких обмежень по пам'яті мобільних пристроїв і більш низькій швидкості процесора;
- облік особливостей екранів і екранних навігаторів конкретних моделей мобільних телефонів;
- сумісність с основними форматами файлів: .doc, .ppt, .pdf, .jpg та ін.;
- мультимедійні можливості: малюнки, відео, аудіо, обмін мультимедійними повідомленнями;
- підтримка комунікаційних та мережевих технологій: Wi-Fi/WiMAX, Bluetooth, GPRS, EVDO, GSM, CDMA [3].

Метою роботи є дослідження принципів побудови платформи Android та перспективи її подальшого розвитку.

Android - це мобільна операційна система, яка заснована на зміненому ядрі Linux, і використовується для мобільних телефонів, смартфонів, комунікаторів, планшетних комп'ютерів, електронних програвачів і інших мобільних пристроїв [1].

До особливостей платформи Android слід відности:

- Різноманітність моделей. Пристрої на основі Android вже пройшли на ринку початковий етап становлення і тепер користувач може підібрати для себе симпатичні й елегантні моделі пристроїв, що підкреслюють їх індивідуальність, до того ж на ринку електронних пристроїв з даною платформою можна знайти не тільки моделі з захмарними цінами, а цілком доступні для середнього класу населення.
- Відкритість системи. Android представляє собою відкриту операційну систему, що на практиці це дозволяє застосовувати величенний спектр ігор і додатків, у яких чудова графіка при цьому інтерфейс залишається простим і зручним у повсякденному житті.
- Швидкість роботи. У порівнянні з іншими платформами, Android – працює напрочуд швидко, тут не доведеться стикатися з такими проблемами як -зависання в самий невідповідний момент, несподівані самовільні перезавантаження. Сенсорна система дисплея на Android реалізована досконало, тому легким рухом пальця можна скасувати ненавмисні дії, без наслідків.
- Вдосконалене обладнання та аксесуари. Армію прихильників Android також приваблює і те, що тут відмінна якість фото, відео, мультимедіа. Подобається і підтримка великої кількості форматів, що, безумовно, є плюсом.
- Персоналізація. Сучасні онлайн програми на Android тісно прив'язані до інтернету, а значить, програми постійно оновлюються, синхронізуються. Тобто на відміну від інших платформ, пристрій налаштовується під користувача, так що працювати з Android дуже зручно, і кожен користувач впевнений, що даний пристрій створено саме для нього. А кращого й бажати не доводитися [4].

Дуже примітним є той факт, що найменування кожної нової версії ОС Android, є похідною від найменувань різноманітних десертів. Перші букви кожного нового продукту відповідає літерам з латинського алфавіту. Кожна остання версія включає новітні опції і вирішує попередні недоробки [2].

Починаючи з версії 3.1 оновлення виходять раз на 6 місяців. На даний момент випущено 10 версій системи . Виходячи зі статистики за підсумками вересня 2013 року, частки версій розподілилися наступним чином:

Таблиця 1 – Статистика популярності різних версій ОС Android

Версія ОС	**Кодова назва**	**API додатки**	**Поширення**
2.2	Froyo	8	2.2%
2.3.3-2.3.7	Gingerbread	10	28.5%
3.2	Honeycomb	13	0.1%
4.0.3-4.0.4	Ice Cream Sandwich	15	20.6%
4.1.x	Jelly Bean	16	36.5%
4.2.x		17	10.6%
4.3		18	1.5%

На даний момент відносною більшістю володіє версія 4.1.x Jelly Bean (з липня 2013 року) . До цього протягом майже двох років (з жовтень 2011 по липень 2013 року) домінувала 2.3.3 - 2.3.7 Gingerbread .

До безумовних переваг операційної системи Android можна віднести:

- Qwerty - екранна клавіатура;
- документи та пошта доступні в будь-який час;
- більше 70 тисяч програмних продуктів;
- відео- , фото- та аудіо-редактори;
- солідне число розважальних додатків;
- висока продуктивність: завантаження фото і відео за мить, повні фільми онлайн проглядаються без затримок;
- сервіси Google: YouTube, Maps, Gmail, Gtalk;
- спрощений і швидкий вихід в мережу Інтернет;
- доступний інтерфейс, що налаштовується;
- доступ до соціальних мереж [5].

Чого ж можна очікувати в майбутньому від ОС Android? На це питання не можна дати точної відповіді, проте вже зараз ясно, що Google всерйоз взялися за цей проект і по видимому збираються продовжувати активний розвиток. На даний момент вони вже створили конкурентоспроможний продукт, який не обділило увагою жодне велике видання про мобільних телефонів і КПК. Компанії Google можна пишатися тим, що її операційна

система встановлена на більш ніж 1 млрд. мобільних гаджетів у світі. А за п'ять років було випущено понад 11 тис. різних моделей Android-пристроїв. Цифра сама по собі нездорово велика, але саме за різноманітність форм-факторів, техн$_{іч}$них характеристик і ціни люблять «зелених роботів».

Список літератури

1. Голощапов А. Google Android: программирование для мобильных устройств. — СПб.: БХВ-Петербург, 2010.
2. http://www.androidtalk.ru
3. Бачурина А.Ю. Мобильные операционные системы // Молодёжь и наука: Сборник материалов VIII Всероссийской научно-технической конференции студентов, аспирантов и молодых учёных, посвященной 155-летию со дня рождения К. Э. Циолковского [Электронный ресурс]. — Красноярск: Сибирский федеральный ун-т, 2012.
4. http://netgrad-service.ru/index.php/polezniestaty/textproplanshety/388-5----android
5. http://mir-android.com/

УДК 004.75

А.О. Демішонкова
Науковий керівник — Мелешко Є. В., канд. техн. наук, доцент
Кіровоградський національний технічний університет

Сфера застосування хмарних технологій та перспективи їх подальшого розвитку

Суть концепції хмарних обчислень полягає в наданні кінцевим користувачам віддаленого динамічного доступу до послуг, обчислювальних ресурсів і додатків (включаючи операційні системи та інфраструктуру) через Інтернет. Розвиток сфери хостингу був обумовлений потребою в програмному забезпеченні і цифрових послугах, якими можна було б керувати зсередини, але які були б при цьому більш економічними і ефективними за рахунок економії на масштабі.

Метою роботи є дослідження концепцій хмарних обчислень.

У відповідності з темою та метою роботи було сформульовано основні наукові задачі:

– дослідити технологію хмарних обчислень;
– дослідити перспективи розвитку хмарних технологій.

Більшість сервіс-провайдерів пропонують хмарні обчислення у формі VPS-хостингу, віртуального хостингу, і ПЗ-як-послуги (SaaS). Хмарні послуги довгий час надавалися у формі SaaS, наприклад, Microsoft Hosted Exchange і SharePoint [1].

Обчислювальні хмари складаються з тисяч серверів, розміщених в датацентрах, що забезпечують роботу десятків тисяч додатків, які одночасно використовують мільйони користувачів. Неодмінною умовою ефективного управління такою великомасштабною інфраструктурою є максимально повна автоматизація. Крім того, для забезпечення різних видів користувачів: хмарних

операторів, сервіс-провайдерів, посередників, IT-адміністраторів, користувачів додатків захищеного доступу до обчислювальних ресурсів, – хмарна інфраструктура повинна передбачати можливість самоврядування і делегування повноважень [1].

Хмарні обчислення - це ефективний інструмент підвищення прибутку і розширення каналів продажів для незалежних виробників програмного забезпечення (ISV), операторів зв'язку і VAR-посередників (у формі SaaS). Цей підхід дозволяє організувати динам$_{іч}$не надання послуг, коли користувачі можуть проводити оплату за фактом і регулювати обсяг своїх ресурсів залежно від реальних потреб без довгострокових зобов'язань [2].

На думку компанії Parallels, що займається розробкою ПО в сфері хостингових послуг, у найближчі 5-10 років велика частина IT переміститься в хмари п'яти різних типів. Будуть пропрієтарні платформні хмари, що мають різні платформні послуги, - Google (тип 1), Microsoft (тип 2) та інші великі IT компанії (тип 3), такі як IBM, Apple, HP і Amazon [5].

Будуть хмари послуг типу 4, де очікується виникнення тисяч хмарних провайдерів, що пропонують широкий спектр послуг. Як приклад можна навести веб-хостинг і хостинг додатків, вертикально інтегровані структури (уряд, охорона здоров'я, тощо), незалежних виробників ПЗ (стратегічний розвиток бізнесу, системи клієнтської підтримки), телекомунікаційні послуги (голосова пошта, IP-телефонія). І нарешті будуть хмари, керовані корпоративними IT (тип 5), які надаватимуть послуги для внутрішнього використання і для використання співробітниками та партнерами [5].

При сьогоднішньому рівні конкуренції на ринку IT запорукою успіху є перехід до п'ятого типу хмар або залученню сторонніх ресурсів для переходу на четвертий тип. Для вирішення цього завдання Parallels створює рішення, екосистеми і налагоджує партнерські зв'язки з сервіс-провайдерами та компаніями, щоб вибудувати ефективну інфраструктуру надання хмарних послуг. Крім того, Parallels продовжує займатися розвитком SaaS напрямків, щоб забезпечити незалежним виробникам ПЗ та сервіс-провайдерам можливість надавати SaaS-додатки, що відповідають сучасним стандартам галузі [5].

Дослідивши види хмар та їх принцип роботи, можна зробити висновок, що хмарні обчислення є ефективним та принципово новим методом, який дозволяє економити дисковий простір ЕОМ, досягаючи необхідних для користувача результатів. Також це економічно вигідно для хостингів, які мають можливість запропонувати хмарні обчислення користувачам.

Використання хмарних технологій вже впровадилося до широкого застосування і їх подальше використання набере ще більших обсягів. Дана технологія є перспективною завдяки своїй простоті та глобалізації інтернету і вже у найближчому майбутньому використання хмарних обчислень буде таким же розповсюдженим методом, як і спілкування через соціальні мережі або інтернет-торгівля.

Крім того, хмарні технології використовуються часто для створення надзвичайних пристроїв. Так, в Microsoft Research працюють над створенням розумного помічника пам'яті, який підкаже, наприклад, де раніше користувач зустрічав ту чи іншу людину. Розумний помічник пам'яті зможе розпізнавати зображення людей та їх голос, порівнювати їх з уже баченими та надавати

інформацію користувачу системи про те, хто ця людина та при яких обставинах він її зустрічав. Це можливість, яку новий світ пов'язаних пристроїв і хмарні технології зроблять реальним [6].

Задачі забезпечення цілісності інформації у випадку застосування окремих "хмарних" додатків, можливо вирішити завдяки сучасним архітектурам баз даних, системам резервного копіювання, алгоритмам перевірки цілісності та іншим індустріальним рішенням. Та це ще не все. Нові проблеми можуть виникнути у випадку, коли мова йде про інтеграцію декількох "хмарних" додатків від різних постачальників [5]. Саме цей напрямок повинен найближчим часом змінити уставлений статус-кво в інформаційних технологіях. Очікується, що cloud computing спричинить ще більший розвиток Інтернету. Дослідницька компанія Gartner пророкує, що тенденція буде сформована остаточно уже на протязі найближчих декількох років [5].

Також аналітики відмічають, що технологія хмарних обчислень знизить витрати і підвищить попит на нові IT-продукти, однак ефект росту таких технологій проявиться тільки в довготерміновій перспективі [5].

Список літератури

1. Клементьев И. П. Устинов В. А. «Введение в Облачные вычисления» – УГУ, 2009. - 233 с.
2. Определение Облачных Вычислений - Рекомендации Национального Института Стандартов и Технологий (США), NIST, USA, 2007
3. Что такое облачные вычисления и как их можно использовать? - Корпорация IBM, 2008
4. Gillam, Lee. Cloud Computing: Principles, Systems and Applications / Nick Antonopoulos, Lee Gillam. — L.: Springer, 2010. — 379 p.
5. Інтернет-сайт IT-довідок http://habrahabr.ru
6. Офіційний веб-сайт компанії Майкрософт http://www.microsoft.com

---

УДК 004.072.5

О.В. Тріщ
Науковий керівник — Дóренський О.П., викладач, мол. наук. співроб.
Кіровоградський національний технічний університет

Дослідження сторінкової організації пам'яті в комп'ютері

Однією з основних технологій реалізації віртуальної пам'яті в сучасних операційних системах є сторінкова організація пам'яті.

Віртуальний адресний простір кожного процесу поділяється на частини однакового, фіксованого для даної системи розміру, що називають віртуальними сторінками. Вся оперативна пам'ять комп'ютера також поділяється на частини такого ж розміру, що називають фреймами (блоками). Розмір сторінки вибирається рівним степеням двійки: 512, 1024 і т.д., з метою спрощення алгоритму перетворення адрес. При завантаженні процесу частина його віртуальних сторінок розміщається в оперативній пам'яті, а інші - у дисковій.

Сторінкова організація пам'яті повинна мати апаратну підтримку. Вперше вона була впроваджена в процесорах intel 80386.

Кожна адреса, яку генерує процесор, ділиться на дві частини: номер сторінки і зсув сторінки. Номер сторінки використовують як індекс у таблиці сторінок.

Таблиця сторінок – це структура даних, що містить набір елементів, кожен із яких містить інформацію про номер сторінки, номер відповідного їй фрейму фізичної пам'яті та права доступу.

Номер сторінки використовують для пошуку елемента в таблиці. Після його знаходження перевіряються права доступу, якщо їх не має виникає помилка доступу, якщо права є то до базової адреси відповідного фрейму додають зсув сторінки, чим і визначають фізичну адресу рис.1.

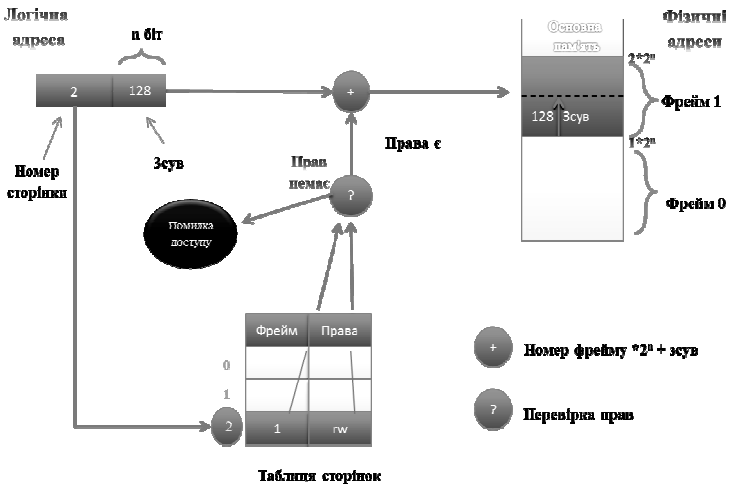

Рисунок 1 – Перетворення адреси в сторінковій організації пам'яті

Для кожного процесу створюють його власну таблицю сторінок. Коли процес починає своє виконання, ОС розраховує його розмір у сторінках і кількість фреймів у фізичній пам'яті. Кожну сторінку завантажують у відповідний фрейм, після чого його номер записують у таблицю сторінок процесу.

Відображення логічної пам'яті для процесу відрізняється від реального стану фізичної пам'яті. На логічному рівні для процесу вся пам'ять зображується неперервним блоком і належить тільки цьому процесові, а фізично вона розосереджена по адресному простору мікросхеми пам'яті, чергуючись із пам'яттю інших процесів. Номер сторінки використовують для пошуку фрейма в таблиці (рис. 2). Процес не може звернутися до пам'яті, адреса якої не вказана в його таблиці сторінок (так реалізований захист пам'яті).

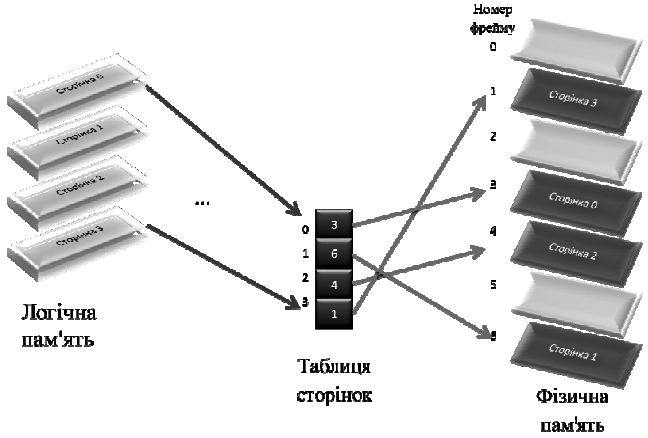

Рисунок 2 – Логічна і фізична пам'ять в сторінковій організації пам'яті

Під час активізації чергового процесу в спеціальний регістр процесора завантажується адреса таблиці сторінок даного процесу. При кожному звертанні до пам'яті відбувається зчитування з таблиці сторінок інформації про віртуальну сторінку, до якої відбулося звертання, якщо дана віртуальна сторінка знаходиться в оперативній пам'яті, то виконується перетворення віртуальної адреси у фізичну, якщо ж потрібна віртуальна сторінка в даний момент вивантажена на диск, відпрацьовується алгоритм сторінкового переривання: процес, що виконується, переводиться в стан очікування, і активізується процес опрацювання сторінкового переривання для пошуку на диску необхідної віртуальної сторінку і завантаження її в оперативну пам'ять. Якщо в пам'яті є вільна фізична сторінка, то завантаження виконується негайно, якщо ж вільних сторінок немає, то вирішується питання, яку сторінку можна вивантажити з оперативної пам'яті.

Щоб адресувати логічний адресний простір значного обсягу за допомогою однієї таблиці сторінок, її доводиться робити дуже великою. Щоб уникнути таких великих таблиць, запропоновано технологію багаторівневих таблиць сторінок. Таблиці сторінок самі розбиваються на сторінки, інформацію про які зберігають в таблиці сторінок верхнього рівня. Кількість рівнів рідко перевищує 2, але може доходити й до 4.

Під час реалізації таблиць сторінок для отримання доступу до байта фізичної пам'яті доводиться звертатися до пам'яті кілька разів. У разі використання дворівневих сторінок потрібні три операції доступу: до каталогу сторінок, до таблиці сторінок і безпосередньо за адресою цього байта. Це сповільнює доступ до пам'яті та знижує загальну продуктивність системи.

Для розв'язання цієї проблеми було запропоновано технологію асоціативної пам'яті або кеша трансляції. У швидкодіючій пам'яті створюють набір із кількох елементів. Кожний елемент кеша трансляції відповідає одному елементу таблиці сторінок.

Тепер під час генерування фізичної адреси спочатку відбувається пошук відповідного елемента таблиці в кеші і якщо він знайдений, стає доступною адреса відповідного фрейму, що негайно можна використати для звертання до пам'яті. Якщо ж у кеші відповідного елемента немає, то доступ до пам'яті здійснюють через таблицю сторінок, а після цього елемент таблиці сторінок зберігають в кеші замість найстарішого елемента.

Сторінкова адресація у Windows XP. Під час роботи з лінійними адресами у Windows XP використовують дворівневі таблиці сторінок. У кожного процесу є свій каталог сторінок, кожен елемент якого вказує на таблицю сторінок. Таблиці сторінок усіх рівнів містять по 1024 елементи таблиці сторінок, кожний такий елемент вказує на фрейм фізичної пам'яті. Фізичну адресу каталогу сторінок зберігають у блоці KPROCESS.

Розмір логічної адреси, з якою працює система, становить 32 біти. З них 10 біт відповідають адресі в каталозі сторінок, ще 10 - це індекс елемента в таблиці, останні 12 біт адресують конкретний байт сторінки (і є зсувом).

Розмір елемента таблиці сторінок теж становить 32 біти. Перші 20 біт адресують конкретний фрейм, а інші 12 біт описують атрибути сторінки (захист, стан сторінки в пам'яті, який файл підкачування використовує). Якщо сторінка не перебуває у пам'яті, то в перших 20 біт зберігають зсув у файлі підкачування.

Логічний адресний простір процесу поділяється на дві частини: перші 2 Гбайт адрес доступні для процесу в режимі користувача і є його захищеним адресним простором; інші 2 Гбайт адрес доступні тільки в режимі ядра і відображають системний адресний простір.

В адресному просторі процесу можна виділити такі ділянки:

- перші 64 Кбайт (починаючи з нульової адреси) — це спеціальна ділянка, доступ до якої завжди спричиняє помилки;
- усю пам'ять між першими 64 Кбайт і останніми 136 Кбайт (майже 2 Гбайт) може використовувати процес під час свого виконання;
- далі розташовані два блоки по 4 Кбайт: блоки оточення потоку (TEB) і процесу (PEB);
- наступні 4 Кбайт - ділянка пам'яті, куди відображаються різні системні дані (системний час, значення лічильника системних годин, номер версії системи), тому для доступу до них процесу не потрібно перемикатися в режим ядра;
- останні 64 Кбайт використовують для запобігання спробам доступу за межі адресного простору процесу (спроба доступу до цієї пам'яті дасть помилку).

Отже, в роботі досліджено принципи на яких ґрунтується сторінкова організація пам’яті, розкриті технології, які пришвидшують обмін даними між логічною і фізичною памяттю.

Список літератури

1. Шеховцов В. А. Операційні системи: Підручник для ВНЗ. / А.В. Шеховцов – К.: Видавнича група BHV, 2008. – 576 с.
2. Таненбаум Э. Операционные системы / Э. Таненбаум. – Питер: СПб, 2002 г. – 1040 с.

УДК 004.38

Р.П. Павлюк
Науковий керівник — Доренський О.П., викладач, мол. наук. співроб.
Кіровоградський національний технічний університет

Аналіз ОС Linux Ubuntu та перспективи її використання для смартфонів

В наш час, еру швидкого розвитку й впровадження інформаційних технологій, вчені та виробники все більше прагнуть полегшити життя людей, тому й з'являються різні пристрої. Ми помічаємо цей розвиток і також у ньому приймаємо участь. Зараз майже кожен має мобільний телефон, в якому розміщені кількаядерні процесори, багато пам'яті і, взагалі, дуже широкі можливості. Але чому ж ми їх не використовуємо? Можливо тому, що для нас це лише телефон, за допомогою якого можна зробити фотографії, перевірити електронну пошту, пограти Angry Birds тощо.

Саме тому голова Canonical Марк Шаттлворт пропонує випустити ОС Ubuntu для телефонів [1], яка зараз використовується на звичайних ПК, тобто в майбутньому розробники телефонів прагнуть замінити комп'ютери.

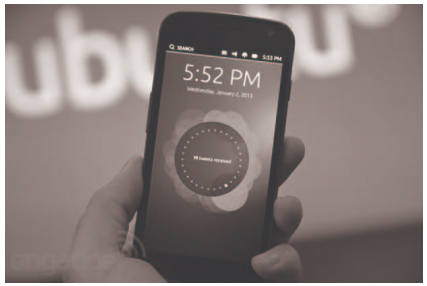

Рисунок 1 – Екран очікування Ubuntu Phone

Ubuntu [2] – це відкрита (безкоштовна) операційна система, яка все більше стає популярною і завойовує увагу користувачів інших ОС (наприклад, Windows). А на мобільних пристроях обіцяє стати взагалі незамінною.

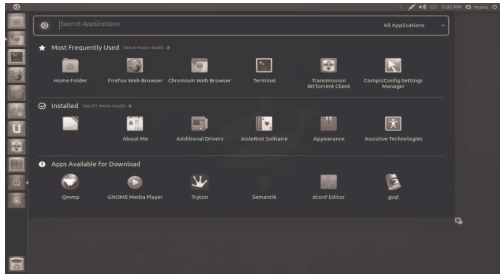
Рисунок 2 – ОС Ubuntu для ПК

Linux Ubuntu буде схожою з ОС Android, а також буде інсталюватись паралельно з ОС Android, яка також на ядрі Linux, тобто користувачам буде запропоновано на вибір більш звичний Android та нова, але досить функціональна, Ubuntu [3]. При цьому дані синхронізуватимуться. Тобто те, що було на Android, буде і на Ubuntu.

Рисунок 3 – ОС Ubuntu на смартфоні

Попри те, що Ubuntu Phone буде дуже зручно підключати до комп'ютера через USB та синхронізувати дані, також можна буде підключати до звичайного монітора та працювати як за комп'ютером [4]. Звичайно, миша й клавіатура також підтримуватимуться.

Рисунок 4 – Підключення до монітора

Отже, як ми бачимо з вище сказаного, оголошена ОС для смартфонів не просто відкриває повні можливості мобільного телефона, але й має великі перспективи в майбутньому повноцінно замінити ПК або ж скласти йому суттєву конкуренцію.

Список літератури

1. В.Костромин, Свободная система для свободных людей (обзор истории операционной системы Linux), март 2005 г.
2. Доступність інформації у сучасному світі на офіційному сайті Ubuntu, автор : 2013 Canonical Ltd. Ubuntu and Canonical are registered trademarks of Canonical Ltd. Перегляд статі : http://www.ubuntu.com/phone.

УДК 004.2

Н.О. Цимбал
Науковий керівник — Доренський О.П., викладач, мол. наук. співроб.
Кіровоградський національний технічний університет

Програмна архітектура смартфона HTC One

Багатьом користувачам знайомі поняття “смартфон” і “комунікатор”, але ринок цих мобільних пристроїв стрімко розвивається і кордони між комунікаторами й смартфонами поступово “розмиваються”. Щороку виходить дуже багато різноманітних мобільних пристроїв. Ось і цьогоріч вийшла велика кількість смартфонів з різними операційними системами (ОС). Дослідивши ринок попиту смартфонів. найпопулярнішими у 2013 році можна вважати Sony Xperia Z, Samsung Galaxy S4, LG G2, Apple iPhone 5S, HTC One.

Метою роботи є дослідження програмної архітектури смартфона HTC One, ядра операційної системи, принцип його функціонування.

Технічні характеристики мобільних пристроїв в силу малих розмірів мають ряд обмежень: маленька потужність процесора, невелика кількість пам’яті, відсутність накопичувачів великої ємності. Всі ці критерії, звичайно, впливають на використовуване в пристроях програмне забезпечення. Саме тому операційні системи в мобільних пристроях повинні відповідати певним системним вимогам. Отже, під час створення будь-якої операційної системи враховується апаратна складова того або іншого пристрою, а розробники програмного забезпечення зобов’язані знати, на якій апаратній архітектурі передбачається використання створюваної ними ОС. На основі цих відомостей відбувається розробка операційної системи.

У лютому цього року в продаж надійшов смартфон HTC One. Хоч він і вийшов на початку 2013 року, проте й досі займає перші місця в рейтингах найпопулярніших телефонів. В порівнянні з попередніми моделями, новий гаджет відрізняється міцним корпусом, стильним дизайном, а також поліпшеною програмною оболонкою, яка зазнала чимало змін.

HTC One є новою флагманською моделлю від тайванської компанії з 4,7-дюймовим Super LCD3 дисплеєм з роздільною здатністю 1920x1080. У HTC One використовується 4-ядерний процесор Snapdragon 600, що працює на частоті 1.7 Ггц, 2 ГБ оперативної пам’яті і 32 ГБ вбудованої флеш-пам’яті для додатків. Смартфон оснащений основною камерою з роздільною здатністю 4 Мп і фронтальною для спілкування за допомогою відеоповідомлень з роздільною здатністю 2.1 Мп. Акумулятор ємністю 2300 мА * год. гарантує тривалий час роботи як при активному використанні, так і в режимі очікування.

Одною з особливостей смартфона є його надпотужний процесор. По-перше , процесорних ядер в ній не два , а чотири. По-друге, працюють вони на частоті до 1,7 ГГц. По-третє, це досить потужні Krait 300. Об’єм кеш-пам’яті другого рівня закономірно подвоївся разом з кількістю ядер – тепер 2 Мб. Snapdragon 600 спілкується з оперативною пам’яттю LPDDR3 по 32-бітовій шині. Графіка – Adreno 320 HF, розширення якого сягає 2048x1536. Тобто в HTC One графіка працює далеко не на межі можливості. МП здатний працювати з камерами, дозвіл яких обмежено відміткою 21 Мп, при цьому можливо робити 3D-фотографії.

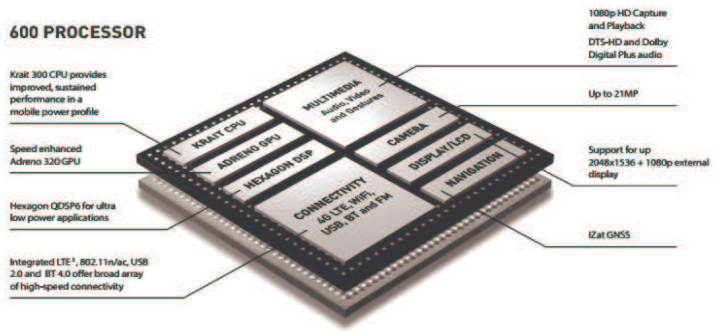

Рисунок 1 – Архітектура процесора Qualcomm Snapdragon 600

Будь-яка операційна система, і Android в тому числі, є набором програм, які керують роботою всього пристрою і відповідають за запуск додатків користувача, таких як ігри, менеджери файлів, веб-браузери тощо.

Ядро Android є, практично, найголовнішою частиною операційної системи, яка відповідає за взаємодію між усім “залізом” і програмною частиною системи. Ядро складається з набору драйверів всього наявного в пристрої обладнання і підсистеми керування пам’яттю, мережею, безпекою та інших основних функцій операційної системи.

Наприклад, коли ви торкаєтеся екрану, щоб запустити додаток, драйвер сенсорної панелі екрана визначає місце, в якому відбулося натискання, і повідомляє координати іншим програмам, що знову ж за допомогою ядра знайдуть в пам’яті пристроя потрібний додаток й запустять його . Це, звичайно, дуже спрощена модель, але суть роботи операційної системи вона відображає.

Рисунок 2 – Принцип роботи ядра Linux

Таким чином, з’ясувалось, що коли будь-яке програмне забезпечення потребує того, щоб обладнання планшета або телефону виконало функцію, воно звертається за цим до ядра операційної системи.

Ядро керує абсолютно всім обладнанням: Wi-Fi, Bluetooth, GPS, пам’яттю і іншими пристроями. Не є винятком і “серце” пристрою – його процесор. Ядро може управляти його частотою і енергопостачанням.

Ядро операційної системи Android запозичене її розробниками, компанією Google, у операційної системи Linux.

Так як ядро керує всім обладнанням, а обладнання у всіх планшетів і телефонів різне, базове ядро Android допрацьовується виробником для кожного пристрою окремо.

Отже, в роботі досліджено архітектуру смартфона HTC One, апаратна платформа, операційна система. Хоч ОС і базується на ядрі Linux, в ній використовуються далеко не всі можливості цієї операційної системи. Причиною тому є використання віртуальної машини Dalvik, в якій і працює все програмне забезпечення. Android має багато недоліків, але розробники усунуть усі недоліки і можливо він стане єдиною довершеною ОС.

Список літератури

1. Snapdragon™: All-in-One Mobile Processor (англ.). Qualcomm. — Краткие спецификации процессоров семейства Snapdragon.
2. Юрий Юрский. Процессоры Qualcomm Krait: четыре ядра, частота 2,5 ГГц.
3. Андрей Коробкин. Платформа Android 4.1 Jelly Bean.[Електронний ресурс] Режим доступу: http://www.4tablet-pc.net/reviews-a-articles/1024-learning-android-what-kernel.html.

УДК 004.4.422

А.О. Демішонкова
Науковий керівник — Бісюк В.А., викладач
Кіровоградський національний технічний університет

Використання “проміжних” представлень програм (CLR)

Common Language Runtime (скорочено CLR – «загальне середовище виконання мов») — це віртуальна машина, яка інтерпретує та виконує код на мові CIL, в який компілюються програми, що написані на NET-сумісних мовах програмування (C#, Managed C++, Visual Basic.Net, Visual J# та ін.), тобто це компонент пакету Microsoft .NET Framework [1].

CLR транслює початковий код в байт-код IL (Intermediate Language) - код на спеціальній мові, що нагадує ассемблер, але написаний для програмної платформи .NET Framework. Саме в нього перетворюється код на інших мовах вищих рівнів, це надає змогу виключити залежність від обраної мови. Реалізація компіляції IL компанією Microsoft називається MSIL, а також надає MSIL-програмам доступ до бібліотеки класів .NET Framework, або так званої .NET FCL (англ. Framework Class Library) [1].

CLR - це той самий механізм, який дозволяє програмі виконуватись в потрібному порядку, викликаючи функції та керуючи даними, тобто CLR керує процесом виконання машинного коду і вирішує яку частину коду куди підставити в певний момент виконання програми [1-2].

Компілятор, крім створення IL-коду, створює повні метадані – набори з таблиць даних, що описують те, що визначено в модулі, тобто таблиці в яких є поля, що вказують який метод належить конкретному файлу та до конкретного типу. Також існують таблиці, що вказують на що зсилається керуючий модуль(наприклад, імпортовані типи та числа). Вони розширюють можливості таких технологій як бібліотеки типів та файлів опису інтерфейсів. Середовище виконання перевіряє, чи доступні всі необхідні ресурси. Використання метаданих дозволяє відмовитися від необхідності зберігати інформацію про компоненти в реєстрі. Отже, при перенесенні компонента на інший комп'ютер більше не потрібно реєструвати цей компонент, а видалення компонента зводиться до простого видалення збірки, яка його містить. Метадані завжди зв`язані з файлом IL-коду, фактично вони вбудовані в *.exe или *.dll [2].

Середовище CLR є реалізацією специфікації CLI (англ. Common Language Infrastructure), специфікації загальномовної інфраструктури, компанії Microsoft [1].

Віртуальна машина CLR дозволяє програмістам забути про багато деталей щодо конкретного процесу, на якому буде виконуватись програма CLR також забезпечує такі важливі служби як:

-керування пам'яттю - автоматичне розміщення об’єктів у комірках пам`яті та керування зсилками на них;

-збірка сміття – вивільнення пам`яті для додатків, методом прибирання об`єктів, час життя яких закінчився;

-керування потоками;

-обробка виключень;

-безпека виконання[3].

Компонент Common Language Runtime розташовується над сервісами операційної системи, якою в даний час є операційна система Windows, але надалі такою може буги практично будь-яка програмна платформа.

Основне призначення CLR – виконання програм, дотримання всіх програмних залежностей, керування пам'яттю, забезпечення безпеки, інтеграція з мовами програмування тощо. Середовище виконання забезпечує безліч сервісів, що полегшують створення та впровадження програм, і істотно покращує надійність останніх [3].

Розробники не взаємодіють з Common Language Runtime безпосередньо: всі сервіси надаються уніфікованою бібліотекою класів, яка розташовується над CLR. Ця бібліотека містить більше 1000 класів для вирішення різних програмних завдань - від взаємодії з сервісами операційної системи до роботи з даними і XML-документами.

Common Language Runtime забезпечує середовище виконання. NET-додатків. Серед наданих цим середовищем функцій слід відзначити обробку виключних ситуацій, забезпечення безпеки, засоби налагодження підтримки версій. Всі ці функції доступні з будь-якої мови програмування відповідно до специфікації Common Language Specification(мінімальний набір можливостей, які повинні реалізувати виробники компіляторів, щоб їх продукти працювали в CLR). Microsoft надає три мови програмування, здатних використовувати CLR:

Visual Basic.NET, Visual C#.NET і Visual C++. Крім того, ряд третіх фірм працює над. NET-версіями таких мов програмування, як Perl, Python і COBOL[3].

Common Language Runtime також задає загальну систему типів, яка використовується всіма мовами програмування. CTS (Common Type System) — є аналогом класу C#. Цей стандарт описує визначення типів та їх поведінку. Також вирішує правила спадкоємності, віртуальних методів, часу життя об'єктів:

- CTS підтримує тільки одиничну спадкоємність (на відміну від C++)
- Всі типи спадкуються від System.Object (Object — ім`я типу, корінь усіх інших типів, System — середовище імен) [3-4].

Всі мови програмування будуть оперувати цілочисельними даними або даними з плаваючою точкою єдиного формату і єдиної довжини, а уявлення рядків теж будуть єдиними для всіх мов програмування. За рахунок єдиної системи типів досягається більш проста інтеграція компонентів і коду, написаних на різних мовах програмування. На відміну від COM-технології, також заснованої на наборі стандартних типів, але які подаються в бінарному вигляді, CLR дозволяє виконувати інтеграцію коду в режимі дизайну, а не в режимі виконання [3].

Мовні компілятори і програми надають функції середовища виконання так, щоб вони були корисні і інтуїтивно зрозумілі для розробників. Можна зробити висновок, що деякі засоби середовища виконання можуть бути помітними в одному середовищі більше, ніж в іншому. Наприклад, розробник Visual Basic при роботі з середовищем CLR може помітити, що мова Visual Basic має більше можливостей об'єктно-орієнтованого програмування, ніж раніше, тому що CLR надає наступні переваги:

-підвищення ефективності розробки додатків;

- можливість легко використовувати компоненти , які розроблені на інших мовах;

-розширені типи , що надаються бібліотекою класів;

-мовні можливості (наприклад, спадкування, інтерфейси і перевантаження) для об'єктно - орієнтованого програмування;

-підтримку явної вільної потокової обробки, що дозволяє створювати масштабовані багатопотокові додатки;

-підтримку структурованої обробки виключень;

-підтримку атрибутів, які можна налаштовувати;

-збірка сміття [4].

Список літератури

1. Вільна енциклопедія Wikipedia [Електронний ресурс] - Режим доступу: www.wikipedia.org
2. IT-довідник [Електронний ресурс] - Режим доступу: http://habrahabr.ru
3. Кевін Бертон NET Common Language Runtime Unleashed 2-volume set/ Kevin Burton. – К.: Sams Publishing, 2002. - 997 с
4. Офіційний веб-сайт Майкрософт [Електронний ресурс] - Режим доступу: http://www.microsoft.com

УДК 004.072.5

Д.В. Шингалов
Науковий керівник — Доренський О.П., викладач, мол. наук. співроб.
Кіровоградський національний технічний університет

Дослідження адресації комп'ютера у захищеному режимі

32-розрядний процесор i80386 відкрив новий етап в історії мікропроцесорів Intel і персональних комп'ютерів типу IBM PC. Природно, він зберігав повну сумісність з своїми 16-розрядними попередниками, щоб не відмовлятися від розробленого для них програмного забезпечення. Але саме в 80386 переборене тверде обмеження на довжину безупинного сегмента пам'яті в 64 Кбайт, що було пережитком минулого і наслідком не самих вдалих архітектурних рішень 8086.

У захищеному режимі 80386 довжина сегмента може досягати 4 Гбайт, тобто весь об'єм фізично адресованої пам'яті. Таким чином, пам'ять фактично стала безупинною. Крім того, 80386 забезпечує підтримку віртуальної пам'яті об'ємом до 64 Тбайт (1 Тбайт = 1024 Гбайт). Убудований блок управління пам'яттю підтримує механізми сегментації і сторінкової трансляції адрес (Paging). Забезпечується чотирьохрівнева система захисту пам'яті і вводу/виводу, а також перекмикання задач.

Процесор 80386, як і 80286, може працювати в двох режимах:

- реальний режим, що цілком сумісний з 8086.
- захищений режим. У цьому режимі можлива адресація до 4 Гбайт фізичної пам'яті (32 розряди), через які при використанні механізму сторінкової адресації може відображатися до 16 Тбайт віртуальної пам'яті кожної задачі.

Перемикання між цими двома режимами в обидва боки, на відміну від 80286, відбувається досить швидко, за допомогою простої послідовності команд, і апаратного скидання процесора не потрібно.

Процесор може оперувати з 8, 16, 32-бітними операндами, рядками байт, слів і подвійних слів, а також з бітами, бітовими полями і рядками біт.

Розрядність регістрів даних (AX, BX, CX, DX) і адрес (SI, DI, BP, SP) збільшена до 32. При цьому в їхньому позначенні з'явилася приставка E (Extended - розширений), наприклад, EAX, ESI. Відсутність приставки в імені означає посилання на молодші 16 розрядів відповідного регістра. Регістри даних і адрес об'єднані в групу регістрів загального призначення, які іноді можуть заміняти один одного. Це може розглядатися як відхід від ідеології спеціалізації всіх регістрів.

Регістри сегментів процесора містять 16-бітні покажчики (у реальному режимі) чи селектори (у захищеному режимі) шести сегментів. З кожним із шести сегментних регістрів зв'язані програмно недоступні регістри дескрипторів, як і у випадку 80286. У захищеному режимі в регістри дескрипторів завантажується 32-бітна базова адреса сегмента, 32-бітний ліміт і атрибути сегментів.

Процесор дозволяє виділяти в пам'яті сегменти і сторінки. Сегменти в реальному режимі мають фіксований розмір, у захищеному - змінний. Сторінки, яких не було в попередніх моделях, являють собою області логічної пам'яті розміром 4 Кбайт, кожна з якої може відображатися на будь-яку область фізичної пам'яті. Якщо сегменти використовуються на прикладному рівні, то сторінки застосовуються на системному.

Процесор 80386 може використовувати режими 32-бітної чи 16-бітної адресації. Режим 16-бітної адресації відповідає режимам процесорів 8086 і 80286, при цьому як компоненти адреси використовуються молодші 16 біт відповідних регістрів.

Так само, як і попередники, процесор 80386 обробляє усі види переривань: апаратні (масковані та немасковані) і програмні, які в даному випадку обробляються як різновид винятків, і власне виняток. Винятки підрозділяються на відмови, пастки й аварійні завершення.

-Відмови (Fault) - це винятки, що виявляються й обслуговуються до виконання команди, що викликає помилку.

-Пастки (Trap) - це винятки, що виявляються й обслуговуються після виконання команди, що викликає цей виняток. До класу пасток відносяться і програмні переривання.

-Аварійне завершення (Abort) - це виняток, що не дозволяє точно установити команду, яка його викликала. Воно використовується для повідомлення про серйозну помилку, таку як апаратна помилка чи ушкодження системних таблиць.

Кожному номеру переривання (0...255) чи винятку відповідає елемент у таблиці дескрипторів переривань IDT (Interrupt Descriptor Table). У захищеному режимі IDT може мати розмір від 32 до 256 дескрипторів, кожний з яких складається з 8 байт.

Відмінності від попереднього процесора 80286 у виконанні операцій вводу/виводу зводяться до додавання можливостей звертання до 32-бітних портів. Важливо відзначити, що рядкові команди процесора 80386 забезпечують блоковий ввід/вивід з більшою швидкістю, ніж стандартний контролер прямого доступу до пам'яті.

Процесор випускався в 100-вивідному корпусі. Була передбачена можливість підключення зовнішнього співпроцесора 80387.

Захищений режим був запропонований для забезпечення незалежності одночасного виконання декількох задач (як системних, так і прикладних). Для цього передбачений захист ресурсів кожної задачі від дій інших задач. Під ресурсами тут розуміється пам'ять з даними, програмами, системними таблицями, а також використовувана задачею апаратура. Захист базується на сегментації пам'яті, причому, на відміну від реального режиму, задача не може перевизначати положення своїх сегментів у пам'яті і використовує тільки сегменти, визначені для неї операційною системою. Сегмент визначається дескриптором сегмента, що задає положення сегмента в пам'яті, його розмір (чи ліміт), призначення і характеристики захисту.

Захист за допомогою сегментації не дозволяє:

- використовувати сегменти не за призначенням, наприклад, трактувати область даних як область програми;

- порушувати права доступу (наприклад, намагатися записувати інформацію в сегмент, призначений тільки для читання, чи звертатися до сегмента, не маючи достатніх привілеїв);

- адресуватися до елементів, що виходять за межі сегмента;

- змінювати дескриптори сегментів, не маючи достатніх привілеїв.

У захищеному режимі передбачаються ієрархічна чотирьохрівнева (рівні 0, 1, 2, 3) система привілеїв, призначена для управління виконанням привілейованих команд і доступом до дескрипторів. Рівень 0 відповідає необмеженим правам доступу і приділяється ядру операційної системи. Рівень 3 дає мінімальні права і приділяється прикладним задачам. Рівні привілеїв відносяться до дескрипторів, селекторів і задач. Крім того, у регістрі прапорів є двохбітове поле привілеїв вводу/виводу , що керує доступом до команд вводу/виводу і прапором переривань.

Механізм віртуальної пам'яті, використовуваний у захищеному режимі, дозволяє будь-якій задачі використовувати логічний простір розміром до 64 Тбайт (16 К сегментів по 4 Гбайта). Для цього кожен сегмент у своєму дескрипторі має спеціальний біт, що вказує на присутність даного сегмента в оперативній пам'яті в даний момент. Невикористовуваний сегмент може бути вивантажений з оперативної пам'яті в зовнішню пам'ять (звичайно - на диск), про що робиться позначка в його дескрипторі. На місце, що звільнилося, із зовнішньої пам'яті може закачуватися інший сегмент (це називається свопінгом чи підкачуванням). При звертанні задачі до відсутнього в оперативній пам'яті сегменту виробляється спеціальний виняток, що і виконує свопінг. З погляду виконуваної програми, віртуальна пам'ять нічим не відрізняється від реальної (говорять, що віртуальна пам'ять прозора), не враховуючи затримки на процес перекачування інформації на диск і з диска.

Реальне використання системи захисту і віртуальної пам'яті покладається на операційну систему, що в ідеалі повинна забезпечувати працездатність навіть у випадку некоректного виконання прикладних задач.

У пам'яті існує три типи таблиць дескрипторів: локальна таблиця дескрипторів LDT, глобальна таблиця дескрипторів GDT і таблиця дескрипторів переривань IDT. Кожній таблиці відповідає свій регістр процесора (відповідно, LDTR, GDTR і IDTR), де зберігаються дескриптори сегментів. Глобальна таблиця містить дескриптори, доступні всім задачам, а локальна може бути для кожної задачі своя. Дескриптори складаються з 8 байтів (як і в 80286). Однак їх призначення різне. Для прикладу на мал.1 показано формати дескрипторів сегмента програм і даних процесорів 80286 і 80386.

Існують також системні сегменти, призначені для збереження локальних таблиць дескрипторів і таблиць стану задач. Їхні дескриптори (теж 8-байтні) визначають базову адресу, ліміт сегмента, права доступу (читання, читання/запис, тільки виконання чи виконання/читання) і присутність сегмента в оперативній пам'яті.

Задачі, дескриптори і селектори мають свої рівні привілеїв. Привілеї задач діють на виконання команд і використання дескрипторів. Поточний рівень привілею задач визначається двома молодшими бітами регістра CS. Привілеї дескриптора описуються полем DPL. DPL визначає найменший рівень привілеїв, з яким можливий доступ до даного дескриптора. Привілеї селектора

задаються полем RPL. Привілеї перевіряються при спробах запису в сегментні регістри, а також при виконанні деяких команд.

Таким чином, починаючи з процесора 80386, з'являються засоби обслуговування багатозадачного режима. Природно, процесор не може обробляти кілька задач одночасно, виконуючи по кілька команд відразу. Він тільки періодично переключається між задачами. Але з погляду користувача виходить, що комп'ютер паралельно працює з декількома задачами.

Список літератури

1. Рудаков П. И., Финогенов К. Г. Программируем на языке Ассемблера IBM PC. — М.: Энтроп, 1996. – С.86-87.
2. Шиндер Д.Л. Основы компьютерных сетей./ Д.Л. Шиндер, С.К.Вильямс, Видавничий центр: АСТ-пресс 2002.-С.234-235.
3. Таненбаум Э. Компьютерные сети, Питер, 2003 –С.123-124.
4. Зубков С.В. Assembler для DOS, Windows и UNIX, Питер-Маркет, СПб, 2004-С.23-25.

УДК 004.4.422

Н.В. Підгорна
Науковий керівник — Бісюк В.А., викладач
Кіровоградський національний технічний університет

Сучасні принципи роботи компіляторів (cloud-технології)

Розподілені обчислення зараз стають основною технологією для побудови великих високопродуктивних систем. В основі всіх хмар лежить віртуалізація - саме вона забезпечує базові характеристики хмари, такі як балансування навантаження, відмовостійкість, паралельні обчислення. Хмарними технологіями називається ціла галузь обчислювальних технологій, сенс якої лежить у віддаленому доступі до обладнання та додатків, встановленим на ньому [1].

Розглянемо характеристики хмарних рішень. У нижній частині хмарного стека знаходиться рівень «інфраструктура як сервіс» (IaaS). Сама інфраструктура тут також перенесена в хмару, яка тепер забезпечує розгортання програмного забезпечення, включаючи бізнес -додатки. Проте рівень IaaS не має середовища для розробки додатків або будь-яких сервісів тестування. Як показано на малюнку, вищим ступенем абстрагування на даному рівні є еластичність, автоматичне розгортання та надання обчислювальних можливостей як комунальних ресурсів [2].

Рівень «платформа як сервіс» (PaaS) надає середовище для створення, розгортання та супроводу прикладного програмного забезпечення. PaaS-провайдер повинен надавати базові сервіси життєвого циклу (збірка,

розгортання, тестування), сервіси для конструктивних блоків (управління станом, транзакції, безпека), а також сервіси для управління ресурсами в процесі виконання [2].

Рівень «програмне забезпечення як сервіс» (SaaS) надає кінцевим користувачам середовище для отримання доступу до додатка і для його використання [2].

Основними хмарними характеристиками, які має підтримувати додаток, є еластичність і мультітенантність. Інші характеристики, такі як ініціалізація і автоматизація, підтримуються за допомогою функцій розгортання сервера додатків і не мають великого впливу на програмний код. Такі вимоги, як паралелізм, розподілене зберігання даних і вдосконалений захист, є допоміжними; вони повинні задовольнятися за необхідності досягнення еластичності і мультітенантності [2].

Розглянемо кожен з цих аспектів більш докладно.

Еластичність - це здатність до масштабування інфраструктури в бік збільшення і зменшення відповідно до потреб. У періоди пікових навантажень до кластеру додаються додаткові примірники, а при зниженні навантаження кількість примірників зменшується. Цей процес повинен здійснюватися динамічно. Дана функція забезпечується можливостями сервера додатків з підтримки технологій динамічної кластеризації.

Еластичність - це не тільки рішення на основі сервера додатків; сам додаток має бути в змозі підтримувати еластичність. Це означає, що програма має бути спроектовано таким чином, щоб вона була здатна оперувати ресурсами, які вона використовує для підтримки паралелізму [3].

Мультітенантність - це властивість, що дозволяє одному екземпляру програми обслуговувати декілька клієнтів; наприклад, якщо п'ять клієнтів використовують сервіс управління контентом, то кожен з цих клієнтів може використовувати один і той же екземпляр додатку з належним поділом даних і параметрів виконання. Для підтримки мультітенантності додаток повинен застосовувати такі механізми, як розподілене зберігання даних, паралелізм, безпеку і слабкий зв'язок. Існує два підходи до підтримки мультітенантності:

- одне фізичне сховище для всіх орендарів («тенант»);
- різні фізичні ресурси зберігання для орендарів [3].

Під паралелізмом розуміється здатність виконувати декілька запитів паралельно або розділяти велику задачу з обробки набору даних на кілька підзадач, що виконуються паралельно. Це покращує використання доступних ресурсів. Застосування паралелізму позитивно позначається на пропускній спроможності і на продуктивності. Підтримка транзакцій забезпечує надійність за допомогою гарантованої синхронізації змін в станах всіх ресурсів. Ці дві концепції знаходяться на протилежних кінцях спектру - якщо ви підсилюєте одну з них, то послабляєте іншу, і навпаки. Належне поєднання паралелізму та

підтримки транзакцій необхідно для балансування цих характеристик, які суперечать одне одному [2].

Слабкий зв'язок гарантує, що кожен виклик сервісу виконується через стандартний інтерфейс; це дозволяє змінювати викликаний компонент і викликає компонент без впливу один на одного. Слабкий зв'язок реалізується за допомогою посередника (proxy), який здійснює виклик. Відсутність збереження стану - це властивість слабкого зв'язку, в якій будь-який виклик сервісу не залежить від попереднього виклику. Цей результат досягається за допомогою управління змінами стану в персистентному сховищі. Обидві ці характеристики є взаємодоповнюючими; вони роблять системні виклики більш вільними від залежностей [3].

Розподілене зберігання даних - це спосіб зберігання даних, при якому конкретне розташування даних не має значення. При цьому підході одні й ті ж дані можуть зберігатися в різних місцях. Цей підхід покращує такі показники, як еластичність і відсутність збереження стану, проте може негативно вплинути на підтримку транзакцій, тому його необхідно застосовувати збалансовано. Чотири стратегії організації розподіленого зберігання даних:

- дубльовані вузли: дані доступні на різних вузлах і негайно реплікуються на інші вузли;
- реплікація на вимогу: задаються тригери, які ініціюють ручну або автоматичну реплікацію даних;
- одностороння реплікація з перемиканням при відмові: є план головних і дочірніх вузлів; при відмові головного вузла обов'язки по реплікації даних передаються відповідному дочірньому вузлу;
- спільне використання файлової системи: ця стратегія застосовується в тих випадках, коли реплікація є дорогою (приклад: ресурси файлової системи) [3].

Безпека хмарного додатки сильно впливає на багато характеристики; такі характеристики, як мультітенантність, паралелізм і слабкий зв'язок, породжують додаткові вимоги до безпеки. Якщо додаток розгорнуто як гібридний (наприклад, має хмарний компонент і локальний системний компонент), необхідно забезпечити "міждоменну" підтримку єдиного входу в систему, що накладає додаткові вимоги на систему безпеки [2].

Отже, ми розглянули основні властивості такі як паралелізм, еластичність, мультітенантність і безпека, завдяки яким можна забезпечити «хмарність» сучасних компіляторів.

Список літератури

1. Gillam, Lee. Cloud Computing: Principles, Systems and Applications / Nick Antonopoulos, Lee Gillam. — L.: Springer, 2010.
2. Шринивас Дж., Рамдас Дж. Extend Java EE containers with cloud characteristics. – 2011.
3. http://ru.wikipedia.org/wiki/Облачные_вычисления.

УДК 004.2

В.О. Даркіна
Науковий керівник — Доренський О.П., викладач, мол. наук. співроб.
Кіровоградський національний технічний університет

Дослідження складових ядра та основних функцій ОС Linux

Linux (також відома як GNU/Linux) – загальна назва UNIX-подібних операційних систем на основі однойменного ядра. Є одним із найвидатніших прикладів розробки з відкритим кодом та вільного програмного забезпечення. Навідміну від власницьких операційних систем, на кшталт Microsoft Windows та MacOS, її вихідні коди доступні усім для використання, зміни й розповсюдження вільно (безкоштовно) [1].

Будь-яка UNIX-подібна операційна система (ОС) складається з ядра та системного програмного забезпечення. Також існують прикладні програми для виконання користувацьких задач.

Ядро є “серцем” ОС. Воно містить файли на запам’ятовуючих пристроях, запускає програми і перемикає процесор, обладнання для забезпечення мультизадачності, розподіляє пам’ять й інші ресурси між процесами, забезпечує обмін пакетами в мережі тощо. Ядро безпосередньо виконує незначну частину загальної роботи обчислювальної системи, але воно надає засоби, що забезпечують виконання основних функцій. Ядро також забезпечую використання прямого доступу до апаратних засобів, надаючи спеціальні засоби для звернення до периферійних пристроїв. Тож ядро ОС дозволяє контролювати застосування апаратних засобів різними процесами, забезпечує захист користувачів тощо. Засоби, надані ядром, використовуються через системні виклики.

Ядро системи Linux складається з наступних основних частин: блок керування процесами, блок керування пам’яттю, драйвери пристроїв, драйвери файлових систем, блок управління мережею.

Найбільш важливими складовими ядра, які забезпечують життєздатність системи, є блок керування пам’яттю й процесами. Він забезпечує розподіл областей пам’яті і swap-областей між процесами, складовими ядра і для кеш-буфера. Блок керування процесами створює нові процеси і забезпечує багатозадачність шляхом перемикання задач.

На найнижчому рівні ядро містить драйвери пристроїв для кожного типу устаткування. Існує досить великий набір різних драйверів, оскільки постійно розробляються нові типи пристроїв. У той же час наявно досить багато однакових пристроїв, які розрізняються тільки тим, як відбувається

взаємодія між пристроєм і драйвером. Така схожість дозволяє використовувати класи драйверів, що підтримують однакові операції.

На рис. 1 наведено загальну архітектуру ОС, яку умовно можна розділити на два рівні, та ядра Linux.

На верхньому рівні знаходиться користувальницький простір (простір прикладних додатків), де виконуються програми користувача. Під користувальницьким простором розташовується простір ядра, де функціонує ядро Linux.

Є також бібліотека GNU C (glibc), яка забезпечує інтерфейс системних викликів для зв'язку з ядром та переходу від прикладних програми до ядра . Це є важливим, оскільки ядро і користувальницький додаток розташовуються в різних захищених адресних просторах. При цьому, в той час, як кожен процес в просторі користувача має власний віртуальний адресний простір, ядро займає один загальний адресний простір.

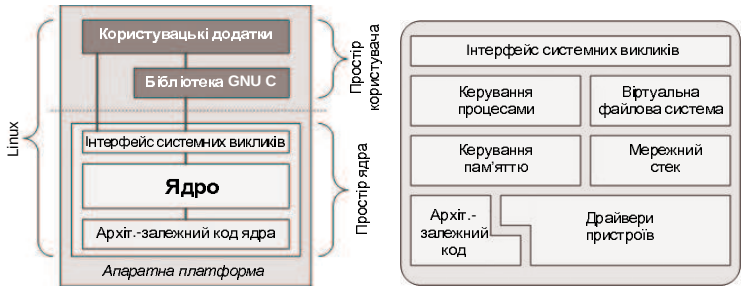

Рисунок 1 – Загальна архітектура ОС (а) та ядра (б) Linux

Слід зазначити, що ядро Linux також можна розділити на три рівня. Нагорі розташовується інтерфейс системних викликів, який реалізує базові функції. Нижче – код ядра, який є загальним для всіх процесорних архітектур, що підтримуються Linux. Останній рівень – архітектурно-залежний код, який утворює BSP (від англ. Board Support Package – пакет підтримки апаратної платформи). Цей код залежить від процесора і платформи для конкретної архітектури.

Список літератури

1. Вікіпедія – вільна енциклопедія: Linux [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Linux
2. Кофлер М. Linux 2013: пер. с англ. / М. Кофлер. – СпБ: «Питер», 2013. – 348 с.
3. Граннеман С. Linux. Необходимый код и команды / Скотт Граннеман. – Спб.: Вильямс, 2010. – 416 с.
4. Лав Р. Linux. Системное программирование / Роберт Лав – Спб.: Питер, 2008. – 416 с.
5. Саров М. ОС Linux: командная строка, утилиты, сценарии: Учебн. пос. / М. Саров – М.: «Вектор ++», 2011. – 25 с.

УДК 004.4

Д.В. Губатенко
Науковий керівник — Сидоренко В.В., д-р техн. наук, професор
Кіровоградський національний технічний університет

Застосування крос-платформового інструментарію Qt для розробки програмного забезпечення мобільних пристроїв

Qt (к'ют) – крос-платформовий інструментарій розробки програмного забезпечення мовою програмування C++ [1]. Він дозволяє запускати розроблене за його допомогою програмне забезпечення (ПЗ) на більшості сучасних операційних систем (ОС) шляхом простої компіляції тексту програми для кожної ОС без зміни початкового коду.

Qt включає в себе всі основні класи, які можуть бути потрібні під час розробки прикладного програмного забезпечення, починаючи з елементів графічного інтерфейсу і закінчуючи класами для роботи з мережею, базами данних, OpenGL, SVG і XML. Бібліотека дозволяє керувати потоками, працювати з мережею, і забезпечує крос-платформовий доступ до файлів.

Сьогодні для того, щоб розробник мав можливість писати програми під різні мобільні платформи йому потрібно володіти декількома мовами програмування і знати різні IDE [2].

Рисунок 1 – Популярні IDE xcode (ліворуч), NetBeans (по центру), eclipse (праворуч)

Тому компанія Digia, яка є власником Qt, в останньому оновленні додала можливість використовувати одну мову програмування та один

інструментарій для розробки програм під усі сучасні мобільні платформи. Для цього потрібно перейти на сайт компанії і завантажити необхідні доповнення. Після їх інсталяції можна відразу починати розробку ПЗ для мобільних платформ. Усі необхідні інструменти знаходяться в тих доповненнях, при чому міняти код не потрібно навіть після встановлення оновлень, оскільки Qt, як вище вказано, є крос-платформовим середовищем розробки. Тому воно автоматично компілює під необхідну платформу, будь то мобільна платформа або на персональний комп'ютер.

Написані програми на Qt не відрізняються ні функціональністю, ні зовнішнім виглядом від програм, написаних на специфічних IDE.

Рисунок 2 – Програми написані на Qt для android (праворуч), та ios (ліворуч)

Метою роботи є ознайомлення з Qt та його можливостями, зокрема:

- крос-платформовість;
- нові можливості.

Також про те, що за його допомогою можна розроблювати програму під різні платформи не змінюючи коду. Звичний інтерфейс та швидкість роботи розроблених програм для різних користувачів. Також для того, щоб почати розробляти програмне забезпечення на Qt, його потрібно просто скачати з офіційного сайту. Це дозволить різним розробникам ПЗ досить легко перейти з інших IDE для конкретних платформ на Qt, в якому дуже легко розроблювати під усі наявні платформи, при чому не змінюючи коду.

Висновки. Крос-платформова бібліотека Qt в найближчому часі може замінити популярні на данний момент IDE для розробки під мобільні платформи завдяки компанії Digia, яка на данний час є власником Qt і працює над її покращенням. Вигляд програм, написаних за допомогою цієї бібліотеки, не відрізняється виглядом і функціональністю від програм, розроблених за допомогою специфічних IDE. Qt є кросплатформовою бібліотекою, за допомогою якої можна розробляти програми для різних ОС, не змінюючи при цьому код програми.

Список літератури

1. Qt 5 для Android: превью/ Eskil Abrahamsen Blomfeldt// Перегляд статті: http:/habrahabr.ru/post/172639/
2. Qt для iOS: превью/ Eskil Abrahamsen Blomfeldt// Перегляд статті: http://habrahabr.ru/post/171739/

Організаційний комітет

Голова

Петренко М.М., канд. техн. наук, професор, перший проректор Кіровоградського національного технічного університету.

Заступник голови

Сидоренко В.В., д-р техн. наук, професор, завідувач кафедри програмного забезпечення Кіровоградського національного технічного університету.

Відповідальний секретар

Дóренський О.П., науковий керівник Студентського наукового товариства Кіровоградського національного технічного університету, викладач кафедри програмного забезпечення Кіровоградського національного технічного університету.

Члени оргкомітету

Якименко Н.М., канд. фіз.-мат. наук, доцент кафедри програмного забезпечення Кіровоградського національного технічного університету;

Даркіна В.О., голова Студентського наукового товариства Кіровоградського національного технічного університету;

Ішуніна Н.М., керівник МОВ Кіровоградського національного технічного університету;

Кава Т.В., фахівець І категорії відділу МОВ Кіровоградського національного технічного університету.

Програмний комітет

Голова

Сидоренко В.В., д-р техн. наук, професор, завідувач кафедри програмного забезпечення Кіровоградського національного технічного університету.

Члени програмного комітету

Паращук С.Д., канд. техн. наук, доцент, завідувач кафедри інформатики Кіровоградського державного педагогічного університету імені Володимира Винниченка;

Смірнов О.А., д-р техн. наук, професор кафедри програмного забезпечення Кіровоградського національного технічного університету;

Мелешко Є.В., канд. техн. наук, доцент кафедри програмного забезпечення Кіровоградського національного технічного університету;

Загорський Р.М., провідний інженер-програміст відділу розробки програмного забезпечення мобільних систем Інтернет-компанії "Онікс";

Змеул О.М., аналітик з програмного забезпечення та інформаційних технологій ТОВ "Вересень Плюс";

Долженко А.А., iOS-розробник Інтернет-компанії "Онікс".