SocioBrains
ISSN 2367-5721, JOURNAL HOMEPAGE: WWW.SOCIOBRAINS.COM
INTERNATIONAL SCIENTIFIC REFEREED ONLINE JOURNAL WITH IMPACT FACTOR
ISSUE 67, MARCH 2020

THE ASYMMETRIC MULTIPROCESSOR OPERATING SYSTEM – A NEW OPPORTUNITY FOR MANUFACTURING OPERATIONS MANAGEMENT

Abstract: An outdated technology in multiprocessor systems can be revived to work with heavy machinery. The Asymmetric multiprocessor operating system is a great opportunity for an irrevocable process and trouble-free operations in manufacturing. The main responsibility of these systems is the basic Input/Output Controllers management.

Author information:

Ekaterina Konstantinova
student
Faculty of Technical Sciences
at Konstantin Preslavsky – University of Shumen
✉ katminkova2@gmail.com
Bulgaria

Tsvetoslav Tsankov
Assoc. prof. Eng., PhD
Faculty of Technical Sciences
at Konstantin Preslavsky – University of Shumen
✉ c.cankov@shu.bg
Bulgaria

Keywords:
Architecture, AsyMOS, asymmetric, multiprocessor, operating system.

1. Въведение

Често срещана тема в изследванията на компютърните науки е важността на I/O устройствата в системи за управление. Един успешен подход е усъвършенстване на самите устройства, прехвърляйки част от функционалността на операционната система (ОС) върху самото устройство, увеличавайки паралелизмът в системата. Въпреки че подобни подходи често осигуряват краткосрочни предимства на производителността, използването на персонализиран хардуер за подобрение на I/O често се оказва и неговият най-голям недостатък, тъй като изостава бързо от увеличаващите се мощности на процесорите с общо предназначение и системните архитектури.

Разработваната в миналото архитектура за асиметрична мултипроцесорна операционна система Asymmetric multiprocessor operating system (AsyMOS), която логически прикрепя процесори с общо предназначение към I/O устройства и по този начин ги правят „по-умни“. AsyMOS работи със SMP системи, които не са обвързани с конкретно хардуерно устройство. При актуализации на архитектурата и на компонентните процесори, обменът на информация ще бъде директен. Те имат много по-изгодно съотношение на цена/производителност от повечето специализирана техника, напр. работна станция, архитектури, водещи до постепенната подмяна в много среди и др.

Различен подход за повишаване на ефективността на устройства е създаден от теорията, че интерфейсите представени от традиционните ОС често крият твърде много от функционалността на устройството зад абстрактни интерфейси на високо ниво, напр. ОС за гнездата – BSD Unix, стандартната система за интерфейси read и write. Въпреки че работят добре при обикновени задачи, те са неподходящи за употреба при системи със стриктни изисквания, напр. аудио/видео плейбек.

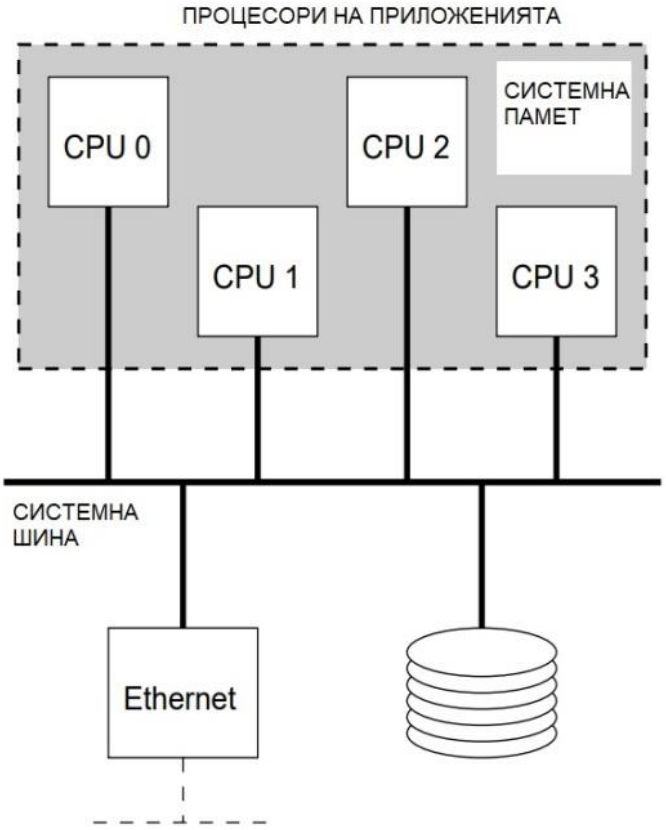
По тази причина, става популярно да се даде степен на свобода на устройствата при използването на техните собствени ресурси. Така ОС осигуряват минимално ниво на функционалност, необходима за споделяне на устройства между множество приложения. Останалите функции, които обикновено се предоставят от ОС, напр. внедряване на файлова система, стек на мрежов протокол, трябва да бъде предоставени от приложенията или споделените библиотеки.

Алтернативен механизъм за осигуряване на конкретни приложения за управление на ресурсите е разширението. Това позволява на приложенията да разширят функционалността на ОС със собствени фрагменти от код. ОС използва различни механизми, които гарантират, че разширенията на различни приложения не могат да въздействат едно към друго, освен при подадена команда.

Архитектурата на AsyMOS осигурява и двата механизъма, така че устройствата да могат да се използват най-ефективно. Вместо даване на достъп само на специфични драйвери, AsyMOS предлага различни функционални интерфейси на ОС за достъп до устройството по най-подходящия начин. Това донякъде е подобно на хардуерните устройства, които предоставят както PIO, така и DMA. Тези функционални интерфейси също имат приложения и директен достъп до „умните“ устройства. И приложенията, и ОС са в състояние да изтеглят разширения на процесорите на устройството, но позволявайки динамичен дял на функционалност между устройството и ОС.

2. Архитектура

На фиг. 1 е показана структурата на традиционна SMP ОС. Група процесори, в случая 4, споделят достъп до голям брой I/O устройства (disk, Ethernet) през системната шина. Процесорите имат достъп до един споделен регион физическа памет чрез памет шина, която обикновено е както по-бърза (тактова скорост), така и по-широка (в битове) от тази на системната шина. В многопроцесорната архитектура на Intel процесорите са изпращали междупроцесорни прекъсвания чрез специална шина.



Фиг. 1. Традиционна SMP ОС

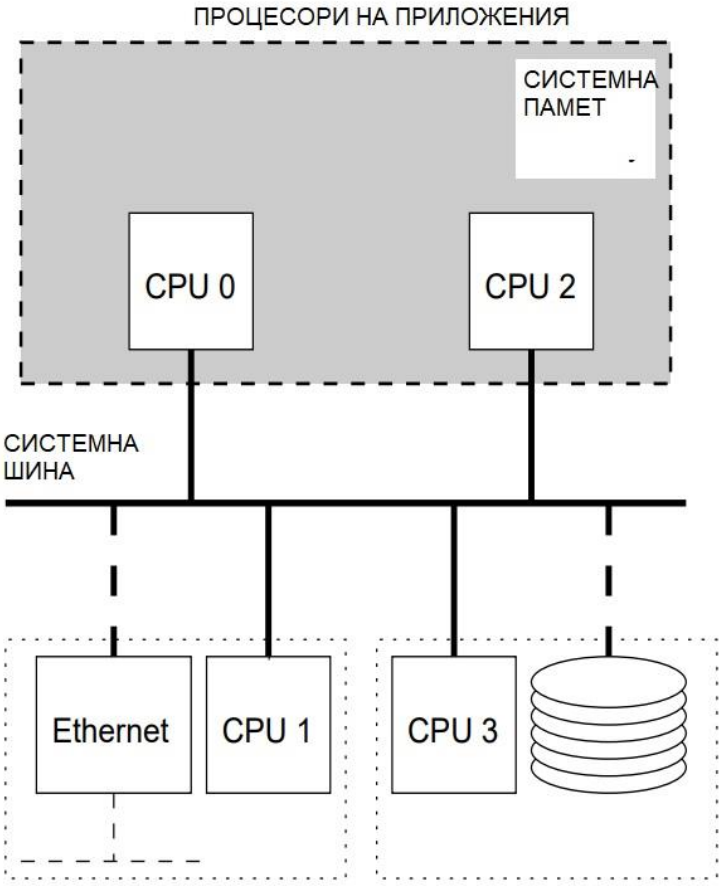
В новите управляващи машини би следвало да се използват предимствата на полупроводниковите запаметяващи устройства SSD. SSD работят най-добре от гледна точка на скорост, износоустойчивост, форма, шум, фрагментация – това са важните фактори. Те не фрагментират файловете, поради липсата на четящи глави, което означава, че информацията може да се съхранява навсякъде. По този начин SSD са много по-бързи [11], [12].

ОС, обикновено стандартна еднопроцесорна, модифицирана за поддържане на множество процесори, приема всички централни процесори като функционално равни. Всяко приложение може да бъде изпълнено от всеки процесор, въпреки че в интерес на ефикасността, ОС може да прикачи приложение към даден процесор. Всеки процесор може да инициира I/O, но обикновено само един обработва прекъсванията [2], [8]. Този подход потенциално осигурява най-ефективно използване на изчислителната мощност, но има два основни недостатъка:

а. Ядрото на ОС трябва да внедри някаква форма на контрол на паралелността на защита на споделените структури от данни. Финият подход е сложен, особено за преобразуване към остаряла еднопроцесорна ОС, така че много системи използват много груб механизъм за защита, намалявайки паралелизма между процесорите;

б. Когато едно приложение осъществява достъп до устройство чрез ОС, кодът на приложението може да бъде изхвърлен от основният кеш принуден от необходимостта да се освободи голямо количество код на драйвера на устройството.

Архитектурата на AsyMOS решава тези проблеми и осигурява други подобрения чрез разделяне на различни процесори на функционални групи (фиг. 2).



Фиг.2 Структура на AsyMOS

2.1. Анализ на AsyMOS

Както е показано обобщено на фиг. 2, AsyMOS разделя процесорите на системата на функционални групи. Тук два процесора се използват като процесори за приложения AP, и два като процесори на устройства DP. DP са разделени на мрежов процесор и дисков процесор. Всеки от тях е свързан с едно или повече устройства, обикновено всички от един и същи клас, напр. мрежа, диск. Комбинацията от устройства и процесори се разпознава от управляващата ОС логически като едно цяло „умно“ устройство.

Една функционална група може да съдържа множество процесори и/или да бъде свързана с множество устройства, напр. един мрежов процесор може да управлява множество Ethernet карти, или два дискови процесора могат да управляват голям масив от дискове. За да се опрости архитектурата на процесора, се поставя отделен процесор, назначен във функционална групата да изпълнява тази задача. Следователно само процесорите, отговарящи за изпълнението на приложения работят на потребителско ниво. Това намалява броя на наличните процесори за изпълнение на изчислителни задачи, но ако работния поток на системата има достатъчно висок I/O дял, тогава цялостното изпълнение ще бъде същото, вероятно е дори да е по-добро, в зависимост от подобренията на AsyMOS [1], [9].

Процесори на приложения и устройства комуникират чрез два механизма – прекъсвания между процесора и споделена памет. Прекъсванията между процесорите осигуряват сравнително закъсняваща комуникация, особено от DP до AP, където AP може да изпълнява произволен код. Тъй като всички процесори имат достъп до паметта чрез високоскоростна

шина, а хардуерът гарантира последователност на кеша, споделената памет осигурява и по-ниска латентност и по-висока пропускателна комуникация. Този механизъм също позволява на приложенията да комуникират директно с AP, без да се налага влизане в основната ОС.

2.2. Предимства на AsyMOS архитектурата

а. Тъй като процесорите на устройства взаимодействат само с устройства и основната ОС, те никога не изпълняват приложения на потребителско ниво, които не се нуждаят от достъп до по-голямата част от функциите на ОС. Вместо това, те използват лекото ядро на AsyMOS – lightweight device kernel (LDK);

б. Целият код за устройството се премества от основната ОС в LDK;

в. Части от функционалността на основната ОС могат да бъдат предадени на процесорите на устройства. Приложенията могат да изтеглят фрагменти на код върху DP по време на изпълнение;

г. Приложенията изтеглят функции динамично на процесора на устройството. Прехвърлянето на функционалността на ОС, което е статично разделение, извършвано, когато основните ОС и LDK се компилират, може да изтегли и фрагменти от код върху DP по време на изпълнение;

д. Процесорът на устройството се справя с всички прекъсвания от свързаните с него устройства. Чрез обединяване на прекъсванията, процесорът на приложения се налага да бъде прекъсван много по-рядко.

2.3. LDK

AsyMOS печели много от своята ефективност пред стандартните ОС поради естеството на LDK. То служи за две цели – обработка на прекъсвания от свързаните с него устройства и комуникация с основната ОС.

а. Тъй като трябва да борави само с устройства от определен клас и общува с основната ОС, то няма нужда от файлова система, терминал, функции за планиране или управление на процесите;

б. LDK винаги работи в най-високо ниво, значително намалявайки разходите за прекъсвач в някои архитектури, напр. Intel Pentium;

в. То не е част от основната ОС, като по този начин премахва нуждата от голяма част от контрола за съвместимост, който възпрепятства паралелизма. Структурите от данни общи както за LDK, така и за основната ОС са модифицирани, за да поддържат сигурен и ефективен едновременен достъп.

И четирите точки взети заедно водят до много по-малък обем на кода за LDK, намаляващ кеша на процесора на устройството и следователно увеличаване на производителността.

Според предвижданията, контролерите с AsyMOS би трябвало да имат много по-добра защита от злоумишлени въздействия, към които са уязвими досега известните контролери [5].

2.4. Техническо изпълнение

Въпреки че прототипът на AsyMOS има само ограничено подмножество от описаните функционалности, той осигурява отправна точка и голяма част от основната технология, необходима за по-пълното им изпълнение. Това първоначално изпълнение се основава на версия 2.0.30 на Linux и работи на двуядрен процесор 166 *MHz* Pentium PC. Процесорът на устройството контролира само една Ethernet карта, 3Com 3c905 100BaseTX бърз Ethernet адаптер. Псевдо устройството, състоящо се от процесора на устройството и тази карта е известен като NetP.

а. Многопроцесорно разпределение на прекъсването е добавено, използвайки Intel I/O Advanced Programmable Interrupt Controller (APIC) за маршрутизиране на прекъсванията към процесори, различни от зареждащия процесор (нормалният режим на работа на Linux);

б. Предвиден е механизъм за предаване на съобщения, които поддържат комуникация между DP и AP. Той използва прекъсвания между процесорите и позволява прости извиквания между процедурите;

в. Планерът е разширен, за да може определени процесори да бъдат посочени като не подлежащи на включване в график. Това средство се използва, за да се предотврати опитът на планерът на Linux да се включи процеси на потребителско ниво.

Тези модификации предоставят рамка, на която се основава внедряването на лекото ядро и NetP псевдо-устройството. LDK понастоящем се реализира като Linux ядро, което обработва прекъсванията и комуникира с процесора на приложението. Използва модифицирана версия на драйвера 3c905 от LDK за управление на Ethernet картата. Извиква функции, които логично са част от Linux ядрото, т.е. онези функции, се превеждат в съобщения от DP до AP.

При управлението на производствени процеси, автоматична идентификация и събиране на данни, двупроцесорни промишлени контролери с асиметрична операционна система могат да имат и много други предимства, освен отказоустойчивост. Така например те могат да бъдат програмирани директно, без да бъдат подложени на опасности от кибератаки. Модифицираната операционна система може да дава различни права на повече групи потребители [4], [6], [10].

3. Резултати

3.1. Прекъсвания, доставени до процесора за приложения

Един от начините, по които AsyMOS се стреми да направи устройства, които изглеждат „по-умни“ е чрез намаляване на прекъсванията:

а. Съчетаване на прекъсвания, така че множество прекъсвания на устройството да се обединяват само в едно известие до процесора на приложението;

б. Обработка на пакети само на ниво устройство. Това е възможно в два често срещани случая: ако драйверът може да определи, че ОС ще отхвърли пакета или ако пакетът е от тип, който може да се управлява изцяло от устройството, напр. ARP, ICMP ping.

Експериментът измерва броя на прекъсванията направени от процесора на приложението поради фоновия трафик в 100BaseTX LAN, за 5 периода всеки с продължителност 5 минути. Резултатите са показани в Табл. 1. Тези резултати ясно показват, че много прост филтър на ниво устройство може да намали броя на прекъсванията с около 80%. При съвременните мрежови адаптери с новите гигабитови мрежи може да има много по-добри постижения.

Таблица 1. Прекъсвания в период от 5 мин

Система	Прекъсвания / 5 мин.
Linux	774,2
AsyMOS	731,8
AsyMOS с филтър	131,0

Както се очаква, скоростите за Linux и AsyMOS по същество са еднакви поради структурата на мрежов код на Linux (прекъсване при всеки „получен пакет“) от картата изпращаща съобщение до ОС за обработка на този пакет.

3.2. Съдържание на кеша съобразено с кодовете на драйверите

Както беше посочено по-рано, другата цел на архитектурата на AsyMOS е да намали съдържанието кеш на приложния процесор. То може да се измери по различни начини, но един

валиден показател е измерване на броя пропуски на I-кеша, направени с цел изпращане на пакет и получаване на отговор [3], [7]. Разликата между производителността на кеш L1 и системната памет е добре изучена и показва неефективното използване на I- кеш паметта като основна причина за лошо изпълнение на протокола за реализиране на стека. Броячът на пропускане на I-кеш се нулира точно преди операционната система да извика hard_start_xmit функцията на драйвера на устройството (задейства драйвера да изпрати пакет) и да прочете функцията netif_rx (за обработка на получения пакет). Кодът между тях представлява следната последователност от действия:

а. Драйверът на устройството изпраща пакет. Картата на 3Com 3c905, използвана в реализацията, инициира DMA прехвърляне – завършването на предаването на пакет се сигнализира с прекъсване;

б. ОС се връща към работещ код на приложението;

в. Прекъсването при „завършването на предаването на пакети“ се обработва от ОС;

г. Пристига пакет с отговора, което води до друго прекъсване.

Драйверът на устройството чете пакета, след което се извиква netif_rx да го обработи. Пакетите, които трябва да бъдат изпратени, са генерирани чрез стандартната програма „ping“ в две конфигурации. Първо, пакетите са генерирани със скорост 1 в секунда, общо 50 пакета. Системата работи на празен ход между чакане на отговор и изпращането на следващия пакет. Второ, 100 пакета са изпратени в „наводняващ“ режим, който активира изпращане на пакет веднага след получаване на отговор от предишния пакет. Това предотвратява връщането на системата в режим на празен ход, макар и не между изпращане и получава, т.е. 48 пакета в първият тест, 98 – във втория. Всеки тест се повтаря 5 пъти и средните резултати, са записани в табл. 2. Те показват, че AsyMOS намалява съдържанието в кеша на с около 20–30%. Това се дължи изцяло на замяната на сложни устройства с по-компактния AsyMOS. Следователно е разумно да се заключи, че AsyMOS ще покаже по-добри резултати спрямо Linux.

Таблица 2. Пропуски на I-кеша за изпратена-получена двойка пакети

Система	Бавен ping	Наводнителен ping
Linux	380,6	355,6
AsyMOS	297,9	250,5

4. Заключение

Измервания на традиционен многопроцесорна UNIX ОС показват, че системата обикновено прекарва около 30–60% от времето, изпълняващо I/O. Освен това компютрите все по-често да се използват в среди, където I/O е по-важно от изчислението, напр. мрежови компютри, сърфиране в мрежата и др. Тези два факта водят до мисълта, че съотношението на натоварването на съвременната система е достатъчно високо, за да направи AsyMOS приложима архитектура.

Тя е особено подходяща за управление на мрежови устройства, които обикновено са сравнително „глупави“ и изискват много грижи и внимание. При отдаване на част от изчислителните ресурси на тези устройства, се премахва отговорността от операционната система, което води до увеличаване на цялостната работа на системата и мрежата.

Може да се твърди, че допълнителните процесори за комуникационна поддръжка в AsyMOS биха били по-ефективни като процесори с общо предназначение. Определението за ефективност трябва да бъде придружено от дефиниция и показатели за натоварване. С архитектурата на AsyMOS се отменят много от основните слабости (сложност, съдържание на

ресурси и т.н.) на еднопроцесорна ОС, напр. Linux, работещ на SMP хардуер, а също така осигурява по-голяма отчетност и контрол на ресурсите. Тази комбинация от предимства ще позволи на комуникационно-ориентираните приложенията, ще бъдат по-ефективно поддържани от AsyMOS, отколкото от симетрична ОС.

С увеличаването на броя на процесорите в мултипроцесорна система проблемите на съдържанието на ресурси и контролът на паралелността стават все по-значими. Следователно AsyMOS е още по-привлекателен от онези системи, при които намаляването на тези ефекти осигурява повишена производителност в сравнение с случая на едно-ядрения процесор. Възможна е по-голяма степен на гъвкавост при настройване на устройството спрямо баланса на процесора към работното натоварване.

Предвид известните кибератаки на съществуващите промишлени контролери с огромните нанесени щети, наложителна е тяхната подмяна с ново поколение контролери, които по никакъв начин да не се поддадат на злоумишлени въздействия. Предимствата на многопроцесорния контролер със специализирана асиметрична операционна система могат коректно да се изброят единствено след реализирането му. В процеса на тестова експлоатация могат да се уточнят и много от рисковете.

References:

1. Bach, M.J., Buroff, S.J., 1995. Multiprocessor UNIX Systems. AT&T, Bell Laboratories Technical Journal, Volume 63, Number 8, Part 2, (October 1984), pp. 1733-1751.
2. Bershad, B. et al., 1995. Extensibility, Safety and Performance in the SPIN Operating System. Proceedings of the 15th ACM Symposium on Operating Systems, Principles, pp. 267-284.
3. Blackwell, T., 1996. Speeding up Protocols for Small Messages. Computer Communication Review, Volume 26, Number 4, pp. 85-95.
4. Boyanov, P., 2015. Detection and implementation of alternate data streams in the computer and network systems, a refereed Journal Scientific and Applied Research (Licensed in EBSCO, USA), Konstantin Preslavsky University Press, vol. 7, ISSN 1314-6289.
5. Boyanov, P., 2019. Identification of active hosts in the computer networks and evaluation the network security against modern types of cyber attacks. International Scientific Online Journal, Publ.: Smart Ideas - Wise Decisions Ltd, Issue 56, ISSN 2367-5721.
6. Boyanov, P., 2019. Implementation of the network vulnerability scanner Armitage for security weaknesses detection in the computer network and systems, a refereed Journal Scientific and Applied Research (Licensed in EBSCO, USA), Konstantin Preslavsky University Press, ISSN 1314-6289.
7. Clark, D., Tennenhouse D., 1990. Architectural Considerations for a New Generation of Protocols. Computer Communications Review, Volume 20, Number 4, pp. 200-208.
8. Davie, B., 1991. A Host-Network Interface Architecture for ATM. Computer Communication Review, Volume 21, Number 4, pp. 307-315.
9. Druschel, P., Peterson, L., Davie, B., 1994. Experiences with a High-Speed Network Adapter: A Software Perspective. Computer Communication Review, Volume 24, Number 4, pp. 2-13.
10. Kazakov, S., Yankova-Yordanova, Y., 2017. Typology of risks in RFID. Journal scientific and applied research, Vol. 12, ISSN 1314-6289.
11. Trifonov, T., 2014. Performance analysis of a mobile computer equipped with solid state disk. Annual of Konstantin Preslavski University of Shumen, ISSN 1311-834X, pp. 27-42.
12. Vasilev, V., Yankova-Yordanova, Y., 2016. Gradation from HDD to SSD. Scientific conference MATTEH 2016, Shumen, ISSN 1314-3921.