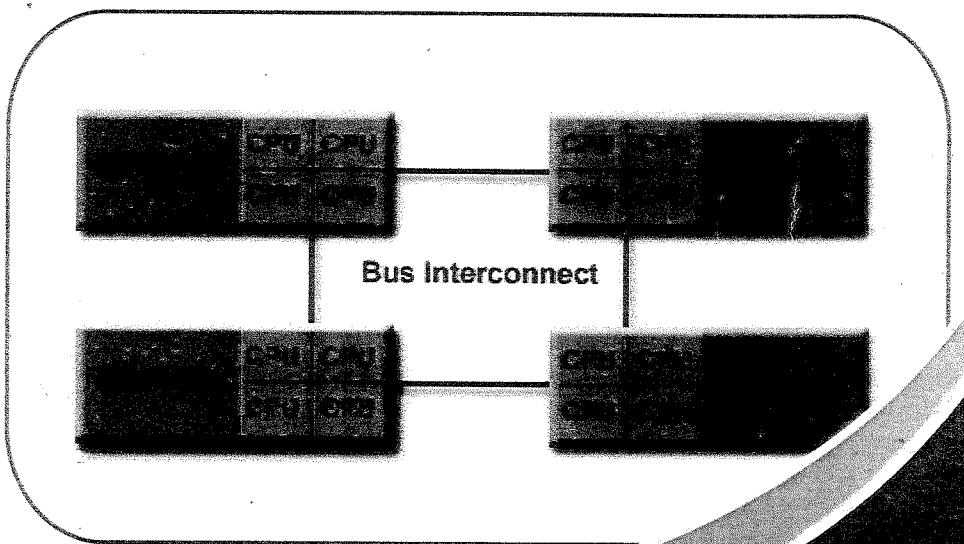
Цветан Таслаков Милен Ангелов

КОМПЮТЪРНИ АРХИТЕКТУРИ

Ръководство за лабораторни упражнения



Технически университет
Варна 2010

ТЕХНИЧЕСКИ УНИВЕРСИТЕТ
ВАРНА

Цветан Таслаков
Милен Ангелов

КОМПЮТЪРНИ АРХИТЕКТУРИ

РЪКОВОДСТВО ЗА ЛАБОРАТОРНИ УПРАЖНЕНИЯ

ВАРНА
2010

ПРЕДГОВОР

Ръководството за лабораторни упражнения по дисциплината "Компютърни архитектури" е предназначено за студентите от специалност "Компютърни системи и технологии". То се явява четвърто преработено и допълнено издание, отчитащо настъпилите промени в учебните планове на дисциплината. Ръководството има за цел подпомогне студентите в усвояването на някои от въпросите, разглеждани на лекции и същевременно да даде нови знания по съответните теми.

Разширяването на знанията по архитектура на компютъра в настоящото пособие е предложено да става чрез моделиране на функциите на различни устройства, включени в компютъра. В първото упражнение се разглеждат основни въпроси, свързани с оценяването на производителността. Всички останали упражнения имат еднотипна структура: ЦЕЛ НА УПРАЖНЕНИЕТО, ТЕОРИЯ, МОДЕЛНИ ИЗСЛЕДВАНИЯ, КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ и ЛИТЕРАТУРА. В ЦЕЛ НА УПРАЖНЕНИЕТО се формулира задачата, която е обект на изследване в упражнението. В частта ТЕОРИЯ се разглежда накратко модела, чрез който се търси решение на поставената задача. Дават се параметри, които той отчита за определяне търсените характеристики, а също така и ограниченията му. В МОДЕЛНИ ИЗСЛЕДВАНИЯ се дават какви зависимости е необходимо да се получат, а също така и стойностите на параметрите, с които се провеждат самите изследвания. В последната част - КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ се поставят въпроси и задачи, на които студентът следва да отговори, ако внимателно е слушал лекциите и е осмислил темата на упражнението.

За всяко упражнение е разработена съответна моделираща програма, в която чрез диалогов режим се въвеждат входните параметри и се получават резултатите във вид на таблици и графики.

КОМПЮТЪРНИ АРХИТЕКТУРИ
ръководство за лабораторни упражнения

Автори: © Цветан Таслаков
© Милен Ангелов

ISBN 978-954-20-0484-4

След всяко упражнение студентите предават отчет (по един от всяка работна група), в който трябва да е отразено: имената на работилите студенти с техните факултетни номера и група, темата на упражнението, входните параметри, с които са проведени изследванията, получените резултати (в табличен и графичен вид) и изводите. Изводите са най-съществената част от упражнението!

Отделните упражнения от ръководството са написани както следва:
упражнения 1,2,3,4,5,6,7,8,9,10,11,12 - от Цветан Таслаков
упражнения 1,2,11,12 - от Милен Ангелов.

Авторите изказват благодарност на всички студенти, участвали в подготовката на моделиращите програми, а също така е на рецензентите доц. д-р Н. Рускова и доц. д-р Д. Тянев, които със своите забележки и предложения спомогнаха за подобряване на изложението и отстраняване на някои неточности.

Авторите

СЪДЪРЖАНИЕ

УПРАЖНЕНИЕ 1

ПРОИЗВОДИТЕЛНОСТ НА КОМПЮТЪРА. МЕТОДИ ЗА ОПРЕДЕЛЯНЕТО Й.

1. ОПРЕДЕЛЯНЕ НА ПОНЯТИЕТО ПРОИЗВОДИТЕЛНОСТ

Компютрите постоянно се усъвършенстват. Главният вектор на тяхното развитие е повишаване на производителността. Този вектор е изключително важен. Съгласно закона на Мур, нямащ строго доказателство, но потвърждаващ се от практиката, производителността на "обикновените" компютри се удвоява на всеки 18 месеца.

Производителността на компютъра, както ще стане ясно по-долу, е тясно свързана с неговата архитектура. Ето защо нейното определяне представлява определен интерес за специалистите по компютърна архитектура. Предварително трябва да се подчертае, че оценката и сравнението й за различни компютри представлява от само себе си сложен проблем, който всъщност не е получил до сега удовлетворително решение.

Под понятието "производителност", в най-общия случай, се разбира (и оценява) количеството извършена работа за единица време. Това определение сега трябва да се конкретизира за компютъра. Самото определение включва два компонента – работа и време. Работа, от гледна точка на компютъра, е изпълнението на дадена програма, а от своя страна то зависи от такива параметри на компютъра, като:

- времето за изпълнение на отделните команди (те пък от своя страна зависят от тактовата честота на процесора и броя тактове за всяка команда);
- логическите възможности на системата команди;
- структурата на процесора (брой регистри, форматът на данните и т.н.);
- паметта (структура на паметта – кеш памет, основна (RAM) външна- обема в МВ на всяка една от тях и времената за достъп;
- пропускателната способност на каналите за външен обмен;
- състава и характеристиките на периферните устройства;
- характеристиките на операционната система,
- типа задачи, стила на програмиране и др.

Времето най-често се дава в секунди или нейните подразделения – мили и микросекунди.

И така производителността на компютъра представлява сложна нелинейна функция от така изброените фактори (параметри).

Отделните параметри, включени в понятието производителност, е прието да се наричат индекси на производителността. Различните индекси на производителността описват различни аспекти от поведението на системата. Ясно е, че някои от индексите са количествени, а други - качествени. Количествените индекси се измерват с дименсионно различни единици. За много от качествените индекси е трудно и дори невъзможно да се даде количествена оценка, например структурността на програмата или езика за програмиране. Ето защо по-нататък ще се разглежда ограничено определение на производителността, което предлагат индексите, лесно приемащи количествена оценка.

Индексите биват първични и вторични в зависимост от това върху кои от тях се акцентува при определянето на производителността. Един и същ индекс за различните компютри може да бъде първичен или вторичен. Например индексът "скорост на изпълнение на операциите" е първичен за суперкомпютрите и вторичен за преносимите компютри.

Различават се клиентска (потребителска) и системна производителност. Клиентската производителност е тази, която е постигната за отделно взета задача, на която е предоставен приоритет за използване на всички необходими ресурси. Системната производителност - това е производителността, достигната при едновременното решаване на съвкупност от отделни задачи на клиента (клиентите).

Най-често под понятието производителност се разбира бързодействието на компютъра (производителността се оценява или изследва по този индекс). В този случай тя се измерва в **MFLOPS** (**M**illion of **F**loating point **O**perations **P**er **S**econd - милиони операции с плаваща запетая в секунда). Понеже съвременните компютри са изключително бързи, все по-често се използват следващите мерни единици – **GFLOPS** и **TFLOPS** (съответно 10^9 и 10^{12}). В някои случаи е по-подходящо да се определя

производителността като пропускателна способност (това е брой изпълнени задания за единица време) или ширина на лентата на пропускане (bandwidth) – това е брой битове или байтове предавани за единица време – **Mb/s** или **MB/s**.

2. МЕТОДИ ЗА ОЦЕНКА НА ПРОИЗВОДИТЕЛНОСТТА

Методите, чрез които може да се получи информация за значението на индексите на производителността, се наричат методи за оценка. Те биват: метод на измерването и метод на моделирането.

Общ проблем на всички методи за оценка на изчислителните системи се явява описанието на работното натоварване. То се определя като множество от входни сигнали (параметри), генерирано от потребителя.

Познатите средства за определяне на производителността принадлежат към една от двете основни групи: методи на измерването и методи на моделирането.

Методите на измерването се прилагат когато:

- компютърът, или негов блок, който се изследва е в наличност (произведен и достъпен);
- в зависимост от това кой индекс на производителността се оценява, са необходими съответните измерителни прибори.

Методите на моделирането за оценка на производителността се прилагат когато:

- компютърът се намира във фаза на проектиране, като целта е предварително да се определят различните му характеристики и целесъобразността на приетите принципи на организация;
- компютърът е произведен, но липсва достъп за провеждане на съответните измервания.

В реалния свят, методите на измерването и моделирането взаимно се допълват, за да може да се получи по-качествена и по-пълна оценка на изследвания обект (компютър).

2.1. Методи на измерването

Към групата "методи на измерването" спадат: измерване на елементарните времена, командни смеси, образцови програми, измервателни програми и синтетични програми.

•**Елементарни времена.** Този подход е възникнал исторически първи, в началото на 50-те години на 20 век, и дава възможност за бързо сравняване на апаратната част на различни компютри и основно на техните процесори. Прието е резултатът да се изразява с брой изпълнени инструкции за секунда **MIPS** (от английски – **M**illion of **I**nstructions **P**er **S**econd - милиони инструкции за секунда). Измерванията се правят за някоя от основните операции, например целочислено сумиране. Понеже в съвременните процесори, устройството за реална аритметика е част от процесора, оценката може да се даде и в **MFLOPS**, ако се определя производителността на базата на команди за реална аритметика.

Елементарните времена се използват за бързи сравнявания. За прилагането на този подход е необходимо да се знаят броя тактове за всяка команда и времетраенето на такта за даден процесор. Предимството на подхода се изразява в неговата простота, а като недостатък може да се посочи, че по никакъв начин не се отчита програмното осигуряване, а както е известно то влияе изключително силно върху производителността. Допълнителни проблеми създава работата на конвейера, а също така не е без значение какъв е форматът на използваните данни. Поради тези причини за по-съдържателни оценки на съвременни компютри следва да се използва някой от другите методи. Този подход дава възможност да се оцени пиковата производителност на процесора, която на практика никога не може да се достигне. Ето защо тя често се нарича теоретична производителност.

Пример: Ако тактовата честота на процесора е 1 GHz и най-често изпълняваната команда се заема 2 такта, каква е производителността на този процесор?

Решение: Времетраенето на такта е:

$$t = \frac{1}{F} = \frac{1}{1.10^9} = 1.10^{-9} = 1ns$$

Понеже командата се изпълнява за 2 такта, времето за изпълнение на командата е $2 \times 1ns = 2ns$. Изхождайки от определението, че производителността е брой команди за единица време, се получава:

$$P = \frac{1}{2ns} = \frac{1}{2.10^{-9}} = 500 \quad MIPS.$$

И така производителността на този процесор е 500 **MIPS.**

В съвременните процесори, с цел повишаване на производителността, често паралелно работят няколко функционални устройства, всяко от които се характеризира със своя производителност - P_i. Тогава общата пикова производителност за процесора се дефинира като:

$$P_{\text{пиково}} = \sum_{1}^{n} P_i ,$$

където **n** е броят на функционалните устройства.

Когато се говори за паралелни компютри, по аналогичен начин може да се определи пиковата или теоретичната производителност, като в този случай **n** е броят на процесорите, P_i е пиковата производителност определена за един процесор.

•**Командни смеси.** При използването на командна смес се оценява средното време за изпълнение на различни команди, което повече съответства на практическото използване. Командната смес се формира по пътя на анализ на честотите за изпълнение на различните видове команди при изпълнението на програмата за решаване на достатъчно широк клас задачи. На основата на такъв анализ на отделните команди (операции) се присвояват определени тегловни коефициенти. Например, командната смес на Гибсън, служеща за определянето на производителността на компютъра при решаването на научно-технически задачи има тегловни коефициенти, поместени в таблицата по-долу.

При използване на командни смеси производителността на компютъра се определя по формулата:

$$P = \frac{\sum_{1}^{n} k_i}{\sum_{1}^{n} k_i * t_i}$$

където: k_i е тегловният коефициент на съответната команда;
t_i е времето за изпълнение на **i** -тата команда.

Този подход се стреми да отчете (в неявен вид) програмното осигуряване. Но определянето на тегловните коефициенти се оказва до известна степен субективно и е необходимо щателно отчитане на всички нюанси в системата команди за различните компютри. Тук трябва да се има в предвид, че архитектурата на процесора влияе върху оценката на тегловните коефициенти, така че използването на този подход при кардинално различни архитектури може да доведе до значителни грешки.

Операции	Тегловен коефициент
Сумиране (изваждане) с фиксирана запетая	0,33
Умножение с фиксирана запетая	0,006
Деление с фиксирана запетая	0,002
Сумиране (изваждане) с плаваща запетая	0,073
Умножение с плаваща запетая	0,04
Деление с плаваща запетая	0,016
Логически операции	0,017
Безусловни преходи	0,175
Условни преходи	0,175

•**Образцови програми.** Образцова програма е типична програма, която може да бъде изпълнена от компютър, но се изпълнява на хартия. В този случай

$$производителност = \frac{брой\ команди\ в\ програмата}{време\ за\ изпълнение\ на\ програмата}$$

Времето за изпълнение на програмата се определя на основата на оценките за времената за изпълнение на командите, давани от производителя. При използването на разклонени или циклични

образцови програми е необходимо графът на програмата по съответен начин да бъде представен като линеен такъв. Тези преобразувания са достатъчно трудоемки и много често изискват използването на специални обработващи програми.
Този подход дава добри резултати, защото отчита както апаратните средства, така и програмните (напр. стила на програмиране). По този начин е възможно до голяма степен да се отстранят недостатъците на предходните два подхода. Той трябва да се прилага винаги, когато не е възможно прилагането на следващите два начина за определяне на производителността.

• **Измервателни програми.** Измервателна програма – това е реална програма, която се изпълнява на изследвания компютър. Всяка приложна програма може да се избере като измервателна. За целта е необходимо да се направи обръщение към таймера в началото и в края на реалната програма. Това става чрез използване на функция за обръщение към таймера. Разликата между двете обръщения към таймера дава времето за изпълнение на програмата, изразено в брой тактове. Всеки такт е с определена продължителност, която за различните компютри е различна. За да се получи времето в астрономически единици е необходимо тази разлика да се умножи по продължителността на такта. По-долу е показан фрагмент от програма, записана на ***C***, илюстриращ подхода.

```
time1=timer_now();
TEST /* това е измервателната програма*/
time2=timer_now();
Printf ("Execution time is %f sec \n", (time2-time1)*64e-6);
```

В този пример ***timer_now()*** е функцията, осъществяваща обръщение към таймера, а константата **64e-6** (64 μs) е продължителността на такта на таймера (още се нарича времето за инициализация на таймера). В някои среди стойностите на **time1** и **time2** директно се получават в астрономически мерни единици и затова не е необходимо умножението им по „времето за инициализация на таймера".

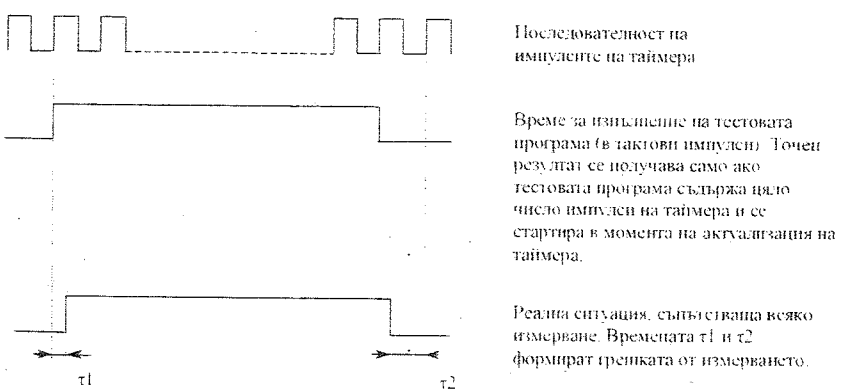
При използване на системния таймер за определяне на времето за изпълнение на една програма се получава приблизителен резултат. Причината е следната – фиг.1-1: Таймерът, с който се

работи, се актуализира през определен интервал (за различните компютри този интервал е различен). В зависимост от това в кой момент се прави обръщение към него и грешката ще бъде различна. Тази грешка не може да се отстрани напълно, но има два начина да се намали.

а) Тестовата програма трябва да реализира обработката на големи структури от данни (масиви).

б) Изчисленията да се повтарят многократно. За целта цялата тестова програма се включва в цикъл, чийто повторения са известни предварително.

Измервателните програми са пригодени за оценка на производителността както на апаратното, така и на програмното осигуряване, при това даже и при сложна операционна среда. Друго предимство е, че вероятността за грешки, извършени от човек се намаляват, защото измервателните програми се изпълняват от компютър и измерването на времето се осъществява чрез неговия таймер. Като техен недостатък може да се посочи, че те не са в състояние да помогнат за предсказване на ефекта при предполагаеми изменения. Освен това сравнително точния резултат, който се получава е в сила за задачи, чиято приложна област съвпада или е близка до тази на тестовата програма.



Фиг.1-1. Причини за наличието на грешки при използване на измервателни програми.

•**Синтетични програми.** Синтетичните програми съвместяват в себе си черти от образцовите и измервателните програми. Това са програми, които са специално съставени така, че да изпитат определени възможности на компютъра. Те също така използват таймера за определяне на производителността. Пример за най-разпространени програми са **Whetstones** и **Dhrystones** тестовете. Синтетичните програми са полезни за оценка на развиващи се системи. При появата на нови функционални средства може да се използват синтетични програми за проверка на тяхната работоспособност.

Всичко казано по-горе относно начина на провеждане на измерванията чрез измервателните програми е в сила и за синтетичните програми. Грешката от измерването при синтетичните програми е от същия характер както и при измервателните програми.

Много често измервателните и синтетичните програми са познати с обобщаващото име **benchmark** програми. Пример за измервателна програма, получила широко признание е **LINPACK** пакета. Тази програма е предназначена за решаване на система линейни алгебрични уравнения с плътна матрица с избор на главен елемент по ред. Прост алгоритъм, регулярна структура на данните, значително изчислително натоварване, близко до пиковото – всичко това прави теста изключително популярен. Понастоящем теста **LINPACK** се използва за формиране на списъка на най-мощните компютри в света (**www.top500.org**).

Друга тестова програма, намерила широко приложение, е Ливерморските цикли (The Livermore Fortran Kernels, **LFK**). Първите версии на този тест се е появил в началото на 70-те години на 20 век, и представлява 24 цикъла, всеки от които е ядро на приложна програма, изпълнявана в Ливърморската национална лаборатория в САЩ. Други тестови програми са NAS parallel Benchmarks (**NPB**), **SPEC** пакета, **SandraSoft** и т.н.

2.2. Метод на моделирането

Чрез метода на моделирането се получава оценка от модел, който описва обекта (компютъра) с някаква степен на правдоподобие. В

случая под понятието модел трябва да се разбира определено количество организирана информация за компютъра, която е построена с цел нейното изучаване.

При определяне на производителността на компютъра, в смисъла на казаното по-горе, най-често се прилагат аналитични или имитационни модели.

2.2.1. Аналитично моделиране. Моделният подход за оценка производителността на компютъра се нарича аналитичен, ако моделът е с математическо представяне на системата. Аналитичните модели биват: детерминирани и вероятностни.

Детерминираните модели се оказват полезни в много области на изчислителната математика и техника. Към тях се отнасят машините на Тюринг, мрежите на Петри, графичните модели на програмите и др. Те намират приложение при извеждането на формули за първо приближение и оценка на порядъка на величините на индексите на производителността. Тази категория модели включва модели, оценяващи средните значения на системите, които могат да се считат получени от вероятностните модели с помощта на замяна всеки случаен параметър с неговото средно значение.Основен техен недостатък се явява трудното отразяване на промяната на работното натоварване, наблюдавано в големите изчислителни системи.

Вероятностните модели по естествен начин отразяват някои аспекти на измененията на работното натоварване. За съжаление други аспекти, като ред на появяване на заданията е или невъзможно да се представи в контекста на модела, или тяхното представяне води до изключително големи трудности при решаването му. Вероятностните модели се базират на:

а) теорията на масовото обслужване, която е известна още под името "теория на опашките";
б) марковските или полумарковските процеси.

Главно предимство на аналитичните модели се явява бързата оценка, която дават. Основно се прилагат, когато **benchmark** тестовете е невъзможно или непрактично да се използват. Едно често подценявано тяхно предимство е възможността да се покаже

кой компонент от компютъра с колко оказва влияние върху производителността.

Пример: Модел на Марини
Марини счита, че производителността на процесора основно зависи от времето за изпълнение на късите и дългите операции и от начина и времето за достъп до паметта. Затова той предлага следния прост модел за определяне на производителността на компютъра:

$$P = (0.7 * t_{add} + 0.3 * t_{mult} + 2 * k_m * t_m)^{-1}$$

където:

t_{add} – време за изпълнение на къса команда (събиране) в μs;
t_{mult} – време за изпълнение на дълга команда (умножение) в μs;
k_m – степен на прозрачност на паметта т.е. съвместяване на избора на командата с нейното изпълнението (k_m = 0 при напълно прозрачна памет);
t_m – цикъл на паметта в μs.

Резултатът се получава в **MIPS**.

В този модел се използва идеята всяка команда да има различен тегловен коефициент. В случая командата събиране, която се явява представител на късите (по времетраене) команди има тегловен коефициент 0.7, а командата умножение т.е. дългите команди – 0.3. Също така, по недвусмислен начин се показва, че производителността на процесора (компютъра) е величина обратно пропорционална на времето.

2.2.2. Имитационно моделиране. За изследване динамичното поведение на изчислителните системи, т.е. как те изпълняват своите функции, широко се използва имитационното моделиране. Имитационният модел представлява описание на изследваната система на някакъв език за програмиране. Той включва описание на елементите образуващи системата, структурата на системата, т.е. съвкупността на връзките между отделните елементи; свойствата на средата, в която функционира изследваната система – функциите на системата. Указаната информация има логико-математически характер и се представя като съвкупност от алгоритми, описващи функционирането на системата. Следователно имитационният модел е програма, построена на основата на тези алгоритми, а имитационното моделиране представлява провеждане на експерименти с модела чрез изпълнение на програмата със зададено множество значения на входните данни.

Процесът на имитационното моделиране може да се раздели на следните етапи:

Създаване на концептуален модел. Концептуалният модел е абстрактен модел, подчертаващ причинно-следствените връзки, присъщи на изследваната система. Тези връзки са съществени в рамките на определеното изследване. На този етап на основата на поставената задача се определя общата идея на модела, степента на методиката за провеждане на изследването, програмните и технически средства.

Разработка на имитационен модел. На този етап се разработва алгоритмично описание на концептуалния модел, изразяващо се в точно определяне на параметрите, характеристиките, и критериите за ефективност. Крайната цел на този етап е създаването на моделираща програма. Етапът завършва с контролни изпитания на модела и анализ за неговата адекватност.

Моделиране с използуване на компютър. Целта на този етап е сбор с помощта на модела на статистически данни за поведението на изследваната система и тяхната обработка за получаване на необходимите характеристики на системата. На този етап се планират необходимите симулационни експерименти с модела и съответните набори входни данни. Много често за получаване на достоверна информация на този етап е необходимо използуването на суперкомпютри.

Следва да се отбележи, че поради сложността и не формалността на указаните етапи при създаване на имитационните модели широко се използва методът на последователно уточняване на модела, основаващ се на циклично повтаряне на отделните етапи на имитационното моделиране.

Изследването на динамични системи предполага наличие на механизъм за представяне на времето в модела, т.е. установяване на отношение между реалното физическо време, в което функционира системата и машинното време, в което се извършва моделирането. Представяното време в модела се нарича системно време. То се изменя само при изменение на състоянието на системата, т.е. само при настъпване на определени събития. Следователно, системното време е дискретно и нараства с променливи интервали.

В зависимост от използваните езици и програмни средства за реализация на моделиращите програми могат да се отделят два основни подхода за построяване на имитационни модели:

- Използване на алгоритмични универсални езици за програмиране (FORTRAN, PASCAL, C, C++ и др.). Недостатък на този подход е трудността за програмиране, възникваща поради паралелността на алгоритмите, описваща поведението на сложните изчислителни системи и динамичния характер на имитационните модели.
- Алгоритмични езици за системно моделиране (GPSS, SIMSCRIPT и др.). Тези езици предоставят унифициран набор от специализирани средства и понятия, чрез които лесно и удобно се описва структурата и функционирането на сложни системи. Те притежават също така и средства за сбор и обработка за статистически данни в процеса на моделиране.

2.3. Апробация на модела. Основният въпрос при моделирането е доколко добре моделът представя моделираната система. Само утвърдителен отговор на този въпрос може да даде на резултата от изследването необходимата достоверност. Модел, поведението на който се различава от поведението на моделираната система на практика е безполезен. Но какво е допустимото различие? Това зависи от максималната грешка. Моделът е достатъчно точен, ако значенията на получените от него индекси се отличават от индексите, получени от моделируемата система на величина, по-малка от максималната грешка.

Ако изследваната система съществува (реализирана е), то определянето на точността може да се осъществи сравнително просто. Трудности възникват когато моделируемата система не съществува или не е достъпна за експерименти. Това препятствие може да се преодолее, ако се признае, че моделируемата система в действителност е концептуален модел. От практическа гледна точка да се убедим, че концептуалният модел е коректно транслиран в моделираща програма не е лека задача. Този проблем по принцип подхожда на проверка на коректността на програма и към него могат да се приложат същите методи. В частност могат да се изберат редица тестови случаи, използвайки модела за получаване значения на индекса на производителността и проверка за разумност на резултатите.

Ако точността е неудовлетворителна, то моделът трябва да бъде изменен, а процеса на проверка повторен. Тази операция се нарича калибровка.

Грешките, съпътстващи процеса на моделиране биват:

- грешки от формулировка на модела;
- грешки от решението;
- грешки от задаването на параметрите.

3. СРАВНЕНИЕ МЕЖДУ РАЗЛИЧНИ МЕТОДИ ЗА ОПРЕДЕЛЯНЕ НА ПРОИЗВОДИТЕЛНОСТТА

Всеки от методите за определяне на производителността има свои предимства и недостатъци.

Методите на измерването, ако могат да бъдат проведени, са приложими за всички видове компютри – микро, до суперкомпютрите). Точността на получените резултати е най-висока в сравнение с останалите методи за определяне на производителността. Получаваната грешка е от порядъка на 1-2%. Като най-съществени недостатъци могат да се посочат:

а) изследваната система трябва да е действаща, т.е. не е възможно в процеса на проектиране да се използват тези методи;

б) трудно се внасят изменения в процеса на измерване с цел да се получат оценки за различни индекси на производителността, т.е. гъвкавостта е ниска.

Аналитичното моделиране, както и имитационното моделиране са подходящи за приложения още в процес на проектиране, което се явява едно тяхно предимство. Аналитичните методи са подходящи за изследване на системи управлявани от несложен алгоритъм за работа. За тяхната реализация са необходими познания върху математическите методи и калкулатор (за най-простите модели) или компютри (за по-сложни модели) като резултатите се получават за няколко часа. Грешката е най-висока – около 10%.

За да бъдат приложени имитационните модели е необходим

мощен компютър. Резултатите се получават с грешка 2-3%, но за да се достигне тази грешка е необходимо повече време (за тестване на програмата, която е значително по-сложна от аналитичния модел и за натрупване на достатъчно представителна статистика). Гъвкавостта на този подход е сравнима с методите на измерването.

4. ЕФЕКТИВНОСТ

Методът на елементарните времена дава оценка, определяна най-често в **MIPS** и тази оценка представлява ***пиковата производителност*** на компютъра. Тя не отчита никакви загуби във времето, свързани с дейността на другите компоненти на системата, освен аритметично-логичното устройство. Очевидно тази производителност никога не може да се достигне.

Методът на измервателните програми дават една значително по-реалистична оценка от гледна точка на потребителя, за производителността на компютъра. Тази производителност е прието да се счита като ***реална производителност***, тъй като тя е получена при изпълнението на конкретни задачи и при конкретни условия. Реалната производителност никога не може да превиши пиковата производителност. Отношението на реалната производителност към пиковата се нарича ***ефективност на работата на компютъра*** и се означава с ***E.*** Свой дял в намаляването на ефективността внасят всички устройства в компютъра. Ако при решаването на големи програми (задачи) ефективността на работата на компютъра е величина >0.5, то ситуацията може да се счита за добра. Ако обаче ефективността е много ниска, а задачата трябва да се реши възможно най бързо, трябва да се разбере къде се губи производителността и следователно кое устройство (блок) следва да се замени с по-производителен. За целта е необходимо да се определи:

- състава, принципа на работа и времевите характеристики на аритметично-логическото устройство(а);
- състава, размера и характеристиките по време на паметта;
- структурата и пропускателната способност на комуникационната среда;
- компилатора, създаващ неефективен код;

- операционната система, организираща неефективна работа с паметта, особено с външната.

При паралелните компютри, оценката ***E*** за един и същ компютър се колебае в още по-широки граници, защото реалната производителност е пряко свързана с конкретната решавана задача и използвания алгоритъм. Ако реалната програма по-продължително време използва отделните независими устройства в паралелния компютър (конвейери, функционални устройства, процесори), то ***E*** е по-близко до 1.

Освен тази ефективност, макар и по-рядко, се използва една друга ефективност, която се дефинира като:

$$E = \frac{C_{ком} + C_{екс}}{P},$$

където: **E** – експлоатационна ефективност на компютъра;
P - производителност на компютъра;
$C_{ком}$ - себестойност на компютъра;
$C_{екс}$ - себестойност на експлоатацията му.

Много често, поради трудности при определяне на себестойността на експлоатацията на даден компютър предварително, то оценката на ефективността на компютъра се прави по опростената формула:

$$E \approx \frac{C_{ком}}{P}.$$

Смисълът на така определената ефективност е да се определи каква е себестойността на една единица **MFLOPS**. Нещо повече, съществува награда за компютър с най-добро съотношение цена/производителност – *Gordon Bell Price/Performance Prize.*

В тази връзка е интересно да се посочи, че Грош, още в началото на 50 –те години е посочил, че между компютърните възможности и цената съществува квадратично отношение, а именно:

$$C_{kom} = f(P)^2$$

където: C_{kom} е цената на компютъра и ***P*** е производителността.

Последните изследвания показват, че такова отношение все още

се поддържа, но вътре в отделните класове компютри, но не и между класовете.

МОДЕЛНИ ИЗСЛЕДВАНИЯ

Да се разучат различните **benchmark** програми за определяне на производителността на компютрите, предоставени от ръководителят на упражнението и с тяхна помощ да се оцени производителността им. Да се обърне внимание кои фактори влияят върху грешката за всеки конкретен тест и по какъв начин се намалява тази грешка.

КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ

1. Приведете примери, че производителността зависи действително от гледната точка за оценка.
2. Какви са различията между производителността, ориентирана към потребителя и към системата.
3. Какво следва да се разбира под понятието "модел на компютъра"?
4. Какво представлява пропускателната способност на компютъра и как се определя?
5. От какво зависи достоверността на един модел?
6. Какво е това аналитичен модел? Защо той представлява интерес при оценка производителността на изчислителните системи?
7. Какви са предимствата и недостатъците на детермини-раните и вероятностните модели?
8. Какво представлява имитационния модел?
9. Моделът на Марини към коя група методи за оценка на производителността може да се отнесе?
10. Какви грешки съпътстват всяка измервателна програма? Какви са подходите за тяхното намаляване?
11. Ако тактовата честота на процесора е 750 MHz и той дава четири резултата на такт, колко е неговата пикова производителност?
12. Опишете методика, чрез която могат да се определят тегловните коефициенти на всяка команда.
13. Каква е разликата между моделното и физическото време в имитационните модели?

УПРАЖНЕНИЕ 2

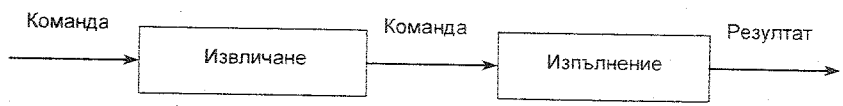
ОПРЕДЕЛЯНЕ ВЛИЯНИЕТО НА КОМАНДИТЕ ЗА ПРЕХОД ВЪРХУ ЕФЕКТИВНОСТТА НА КОНВЕЙЕР ЗА КОМАНДИ

ЦЕЛ НА УПРАЖНЕНИЕТО

Да се изследва производителността на процесор, в който е реализиран конвейерен принцип за изпълнение на командите. Анализът да се проведе с помощта на аналитичен модел.

ТЕОРИЯ

Известно е, че изпълнението на командите включва в себе си няколко етапа. В най-простия случай са два: извличане и изпълнение. В етапа изпълнение има интервали от време, когато обръщения към паметта отсъстват. Тези интервали могат да бъдат използувани за избор на следващите команди. Този подход е илюстриран на фиг.2-1.

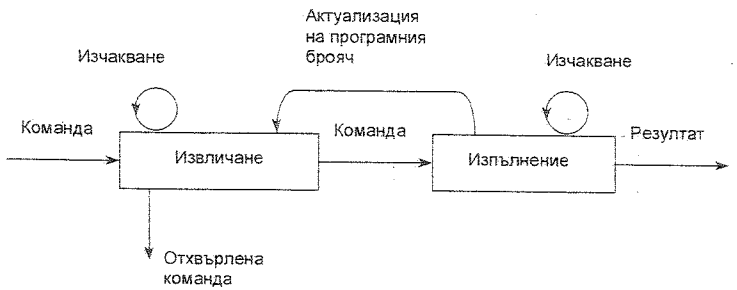


Фиг.2-1. Възможно най-проста структура на конвейер за команди

Така конвейерът има две степени: първата изпълнява подфункцията извличане, втората - изпълнение. Първата степен осъществява избор на командата и нейната буферизация. Когато втората степен се окаже свободна, първата степен прехвърля буферираната команда към втората степен. По време на изпълнението на командата от втората степен, първата степен използува незаетите цикли за обръщение към паметта за избор и буферизация на следващата команда. Описаната процедура е известна под името предварително извличане на команди.

Ако двата етапа имат еднаква продължителност, то времето за изпълнение на командата ще бъде съкратено двойно. Но удвоеното увеличение на скоростта е малко вероятно по две причини:

- Времето за изпълнение на командата, като правило, е много по-голямо от времето за извличане.
- При изпълнение на командите за преход е неизвестно каква команда е необходимо да се чака, докато текущата команда от втората степен се изпълни. След това на етапа на изпълнение настъпва принудително изчакване за избор на следващата команда от паметта. Това е илюстрирано на фиг.2-2.



Фиг.2-2. По-реалистичен модел на двустепенен конвейер

Загубата от време, предизвикана от втората причина може да бъде намалена чрез прогнозиране на следваща команда за изпълнение. Правилото е следното: избира се командата, непосредствено следваща след командата за преход. Ако разклонението не се е състояло, загуба от време не съществува. Ако разклонението се е състояло, командата се отхвърля, а от паметта следва да се избере нова команда.

От изложеното до тук става ясно, че всяка команда за преход води до нарушаване на естествената последователност от постъпващи команди в конвейера. В резултат на това настъпват изчиствания на конвейера и последващо запълване с команди от новия клон на програмата. Това е една от причините, поради която не може да се получи резултат на всеки такт на конвейера. Прието е, когато не се получава резултат на всеки такт, това състояние да се нарича появяване на мехурчета в работата на конвейера. Наличието на множество мехурчета води да спадане на производителността, т.е. **S<<L**, където **S** е коефициента на промяна на производителността, а **L** е броя на стъпалата в конвейера.

С увеличаване на броят стъпала на конвейера този проблем се усложнява. Ето защо при проектирането на конвейера се отделя специално внимание и се предприемат мерки за да се държи конвейера максимално натоварен при изпълнението на командите за преход. Някои от най-разпространените подходи за преодоляване на проблема са:

- предварителен избор на адреса на разклонение;
- използуване на няколко потока команди;
- прогнозирано разклонение;
- отложено разклонение.

Първите три подхода решават проблема на апаратно ниво, а последния – на програмно ниво.

Представлява интерес да се изрази аналитично как влияят командите за преход върху ефективността на конвейера. За целта нека предположим, че броят на стъпалата на конвейера са ***L***, т.е. времето за изпълнение на една команда е ***L*** такта. Очевидно, ако не се появява команда за преход, скоростта на изпълнението ще бъде една команда на всеки конвейерен такт. Да предположим, че общият брой команди в една програма е ***n***. Нека с ***p*** означим вероятността за поява на команда за преход, а с ***g*** - вероятността за успешно взет преход. Според **Amdahl** за една типична програма стойността на ***p*** е около 20%, а на ***g*** - 60%.

Вероятността за появяване на команди, непринадлежащи на командите за преход е ***1-p***. Общият им брой е ***n*(1-p)** и всяка от тях се изпълнява за един такт. Щом за всеки преход, вероятността му за успешното му предсказване е ***g***, то тогава ***n*p*g*** команди, всяка от които също отнема един такт за своето изпълнение, ще отнемат общо ***n*p*g*** такта. Останалите команди - ***n*p*(1-g)*** водят до изчистване на конвейера поради неуспешно предсказване и отнемат ***n*p*(1-g)*L*** такта за изпълнението им. Тогава общият брой тактове за изпълнението на програмата е:

$$n*(1-p)+n*p*g+n*p*(1-g)*L.$$

Ако няма команди за преход в програмата, то нейното изпълнение отнема ***(L+n-1)*** такта. Тогава ефективността е:

$$E = \frac{L+n-1}{n*(1-p)+n*p*g+n*p*(1-g)*L}$$

Ако се предположим, че е изпълнено условието ***n>>L*** (което на практика винаги е в сила), то окончателно се получава:

$$E = \frac{1}{1 - p + p * g + p * (1 - g)L}$$

Действителната производителност **S** се получава по формулата: ***S=L*E***.

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на програма **LAB2** да се проведат изследвания, като се зададат две различни стойности на ***L*** и три различни стойности на ***g***. Получените резултати да се анализират, така че да се дадат отговори на следните въпроси:

1. По какъв начин зависи ефективността на конвейера от вероятността за срещане на команда за преход?
2. Как зависи ефективността на конвейера от вероятността за успешно взето разклонение?
3. Какво е влиянието на дължината на конвейера върху ефективността с отчитане на командите за преход?

КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ

1. Обяснете същността на компенсационните методи, които се използуват в процесорите, за намаляване отрицателното влияние на разклонението:
   - на апаратно ниво;
   - на програмно ниво.
2. Освен при изпълнение на команди за преход при какви други условия може да се наруши ритмичната работата на конвейера, т.е. да не се получава резултат на всеки такт?

УПРАЖНЕНИЕ 3

АНАЛИЗ НА КОНВЕЙЕР ЗА ИЗПЪЛНЕНИЕ НА КОМАНДИТЕ

ЦЕЛ НА УПРАЖНЕНИЕТО

Да се изследва производителността на процесор, в който е реализиран конвейерен принцип на изпълнение на командите. Конвейерът е двустепенен. Анализът да се проведе с помощта на имитационен модел.

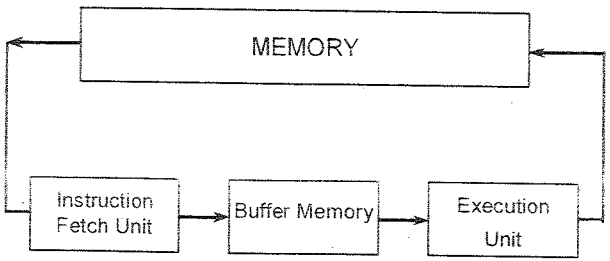
ТЕОРИЯ

Известно е, че техниката „предварително извличане на командата" не води до съществено увеличение на производителността. Причините за това са:

- Времето за изпълнение на командата, като правило, е много по-голямо от времето за извличане. Така времената за изпълнение на подфукциите се различават съществено.
- При изпълнение на командите за преход е неизвестно коя команда трябва да се избере като следваща от паметта.

С цел да се преодолее първия недостатък и същевременно конвейерът да е максимално прост, т.е. да остане двустъпален, в първото стъпало се изпълняват повече действия по цялостното изпълнение на командата, напр. извличане, декодиране, определяне адреса на операнда. Така се осигуряват приблизително еднакви времена за изпълнението на функциите и за двете стъпала. В този случай е прието първото стъпало на конвейера да се означава като **IF**-устройство (**I**nstruction **F**etch **U**nit), а второто степен като **E** -устройство (**E**xecution **U**nit). Между двете стъпала трябва да има фиксатор. Най-простата структура е регистър, който съхранява поредната команда за изпълнение. В този случай връзката е твърда. При запълване на този регистър **IF**-устройството преустановява работата си. По-голямо разпространение е получил фиксатор, съставен от набор от регистри (регистров файл). Така е възможно всяко от стъпалата да работи относително независимо от скоростта на другото.

Силно опростената структурна схема на конвейер за обработка на командите е показана на фиг. 3-1.



Фиг. 3-1. Обобщена структура на двустепенен конвейер за команди

В така дадената структура на конвейера, ресурсите са три: памет, управляващо устройство (**IFU**) и изпълнително устройство (**EU**). Фиксаторът (Buffer Memory), който представлява набор от регистри, работещи на принципа **FIFO**, не се представя като ресурс. Той се счита като опашка към **EU**, което автоматично се организира при провеждане на експериментите на модела.

С цел опростяване на модела се разглежда конвейерно изпълнение на командите за **RISC** процесор. В този случай всички команди се разделят на две групи:

• Първа група. В тази група се включват всички аритметични и логически команди. Техните подфукции са: извличане на командата и изпълнение. Те се появяват с вероятност p. С цел уточняване на модела тази група се подразделя на три подгрупи - команди за безусловен преход, команди за условен преход и чисто изчислителни и логически команди. За целите на настоящия модел е прието, че командите за безусловен преход представляват 5% от първата група команди и предизвикват изчистване на конвейера. Това води до намаляване на общата производителност на конвейера. За командите за условен преход се приема, че представляват 15% и че с вероятност 50% извършват същото като тези за безусловен преход. Третата подгрупа - команди за изчисление и логическа обработка, не предизвикват извънредни събития в конвейера. Програмната реализация на този модел дава две възможности - да се отчитат командите за преход и да не се отчитат. Така е

възможно да се изследва тяхното влияние върху производителността на конвейера.

• Втора група: В тази група се включват само команди за четене и запис от/в паметта. Техните подфункции са: извличане от паметта, определяне на действителния адрес и обръщение към паметта. Те се появяват с вероятност $1-p$.

В това управнение се разглежда компютър, в който е включена една обща памет за данни и команди.

Изпълнението на всяка команда може да се разглежда като процес (транзакт), който се състои от активности, съвпадащи съответно с всеки конвейерен такт. При това са валидни следните събития:
E1 - Начало на извличане на команда.
E2 - Край на извличане на командата. Събитието настъпва след време **t1**, което се явява време за изпълнение на активността "извличане на командата". Това време включва времето за достъп до оперативната памет и четене на адресираната клетка.
E3 - Край на изпълнението. Събитието настъпва след време **t2**, което се явява време за изпълнение на командата или време за определяне на ефективния адрес.
E4 - Край на обръщението към паметта. Събитието настъпва след време **t3**, което се явява времето за изпълнение на активността "запис на резултата". Това време включва време за достъп до паметта и запис в адресираната клетка.

За правилното реализиране на модела е необходимо да се поддържат два атрибута за всеки транзакт. Първият атрибут показва групата, към която принадлежи текущата команда. Този атрибут има статичен характер. Вторият атрибут има динамичен характер и показва събитието, в което се намира транзакта.

Работата на конвейера може да бъде представена по следния начин. **IFU** издава заявка за четене на команда от паметта. Ако тя е свободна, се планира настъпването на събитие **E2** след интервал от време **t1**, в противен случай заявката се нарежда в опашка към паметта. По време на извличането се генерира атрибут, задаващ групата на командата. Ако съдържанието на конвейера е обявено за невалидно, поради успешно изпълнение на команда за преход, извлечената вече от паметта команда се

игнорира и флага за нея се снема, така че следващите извлечени валидни команди да могат да продължат нормално изпълнението си по степените на конвейера. В противен случай (съдържанието на конвейера е валидно) се планира събитие **E3** след време **t2** и едновременно с това се планира събитие **E1**. След това се анализира атрибута за тип на командата. Ако тя е за четене/запис от/в паметта се планира настъпването на събитие **E4** след време **t3**. Ако командата е за безусловен преход или е за условен преход и се осъществи случайно събитие с вероятност 50% се изчиства фиксаторът към **EU** и съдържанието на конвейера се обявява за невалидно, ако в момента се извлича команда. В събитие **E4** се освобождава паметта и ако в нейната опашка има заявки се планира тяхното изпълнение.

В така разгледания модел на работа на конвейера се предполага, че не възникват зависимости между командите; опашката, която се получава към **EU** е с неограничена дължина (т.е. фиксаторът е изграден от произволен брой преместващи се регистри). Тези допускания представляват недостатък на модела и определят неговата неточност.

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на програма **LAB3** да се определи как влияе изменението на времената **t1**, **t2** и **t3** върху производителността на конвейера и средното време за изпълнение на командите с изменение на вероятността ***p*** за появяване на команди за изчисление (***p*** се изменя от 0.1 до 0.9). Времената се задават в тактове. Да се разгледат следните два режима на работа на конвейера и едновременно с това се отчита случите за появяване (не появяване) на команди за преход:

а) **t1=t2=t3=t**;

б) **t1=t3=t,** а **t2>t**;

На базата на получените резултати да се отговори на следните въпроси:

1.Какво е влиянието на времето за моделиране върху резултатите?

За целта се променя само времето за моделиране: програмата се стартира с две различни времена за моделиране – едното малко, а другото голямо. Интересуваме се от промяната на резултатите при две последователни стартирания с едно и

също време за моделиране. Кога ще се получат по-големи разлики в резултатите – при малко или при голямо време за моделиране и защо?

2. Какво е влиянието на времето **t2** върху производителността?

3. Какво е влиянието на командите за преход върху производителността за всеки един от режимите за работа?

КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ

1. Какви функции изпълнява **IFU** и **EU**.
2. На какво се дължи зависимостта между командите в конвейера?
3. Какви други ограничения на модела, освен посочените, можете да посочите?
4. От какво зависи вероятността ***p*** за появяване на команди от първата група?

УПРАЖНЕНИЕ 4

ИЗСЛЕДВАНЕ РАБОТАТА НА МНОГОСТЪПАЛЕН КОНВЕЙЕР

ЦЕЛ НА УПРАЖНЕНИЕТО

Упражнението се базира на работата на реален конвейер, в който е реализиран конвейерен принцип за изпълнение на командите. Целта е да се изследва как се влияе производителността на процесора при увеличаване броя на степените, а също така и какво е влиянието на командите за преход.

ТЕОРИЯ

Един съвременен подход за увеличаване на бързодействието е конвейерът да има по-голям брой стъпала, всяко от тях реализиращо различен етап от обработката на командата, напр. извличане, декодиране, определяне адреса на операнда и т.н. Но с увеличаване броят стъпала в конвейера не следва пропорционално увеличение на скоростта на обработка. Причините за това са:

• Във всяко стъпало на конвейера възникват допълнителни загуби на време, свързани с прехвърлянето на данни между фиксаторите и изпълнението на различни подготвителни функции. Нека да се спрем на този проблем малко по-подробно. Да предположим, че имаме двустъпален конвейер с такт на конвейра 100 ns. Да предположим от тези 100 ns, 20 ns се използват от фиксатора на всяко стъпало, а останалите 80 ns от самата логика за обработка на подфукцията. Сега искаме да увеличим броя на стъпалата на 4 – т.е. два пъти, надявайки се така да се увеличи производителността на конвейера също два пъти. За целта разбиваме базовата функция (изпълняваната команда) не на две, а на четири подфукции. Така всяка подфукция ще изразходва (в идеалния случай) 40 ns. Така такта на новия конвейер става 60 ns, т.е. съкращаваме такта на конвейера не два пъти, а 1.67 пъти и от тука няма да постигнем двукратно увеличение на производителността.

• При увеличаване степените на конвейера стремително се увеличава обема на управляващата логика, необходима за

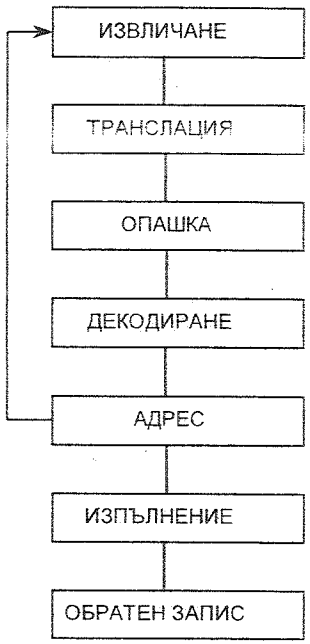
отчитането зависимостите при обръщението към паметта и регистрите, а така също и за оптимизация при използването на конвейера.

• Както стана ясно от упр.2, дългият конвейер повече се влияе от късия конвейер при срещане на команди за преход, в смисъл, неговата ефективност се намалява повече при срещане на команди за преход.

В това упражнение се анализира работата на конвейера, използван в процесора **IDT- C6**, който е съвместим с фамилията **x86** процесори на **Intel**. Интересното в него са използуваните принципи на архитектурната организация. Въпреки, че процесорът изпълнява **CISC** инструкции, то вътрешната му архитектура е ориентирана към тази на **RISC** процесорите. Процесорът притежава 6 степенен изчислителен конвейер за изпълнение на инструкциите. Той не използува наложилата се в съвременните процесори супер скаларна архитектура и произтичащите от нея буфери за пренареждане и допълнителна управляваща логика за това, както и регистри за преименуване. Също така няма и хардуер за предвиждане на преходите. Това дава възможност сравнително лесно (от гледна точка на учебния процес) да се моделира неговата работа.

Конвейерът е подобен на този на **i486** (извличане, декодиране, изчисление на адреса, изпълнение и обратен запис) с изключение на един допълнителен етап - транслация (втори в конвейера). На този етап се транслират сложните **x86** инструкции в по-прости, **RISC** инструкции (в реалният процесор тези по-прости инструкции, които са **RISC** инструкции са 33 битови; а когато броят на RISC инструкциите, с които се заменя всяка x86 инструкция, е по-голям те се извличат от вътрешна **ROM** памет, подобно на другите съвременни **x86**). Така че от гледна точка на програмиста, процесорът е със сложен набор инструкции, а негово изпълнително ядро работи с **RISC** инструкции.

Опростената структура на конвейера на **IDT-C6** е показана на фиг. 4-1.



Фиг. 4–1. Структурна схема на конвейер с преобразуване на сложните команди в по-прости

Концептуалният модел на конвейера разглежда следните 7 ресурса:

- кеш за инструкции;
- блок за транслация;
- блок за декодиране;
- блок за определяне на адреса;
- блок за изпълнение;
- блок за обратен запис;
- кеш за данни.

Опашката преди блока за декодиране не се разглежда като отделен ресурс. Тя се организира автоматично към декодера.

С цел опростяване на модела винаги се предполага, че обръщението към кеш паметта е успешно, т.е. не се извършват

обръщения към оперативната памет, затова тя не се разглежда като ресурс. (Когато се разполага с голяма кеш памет, обръщенията към оперативната памет са редки събития и се предполага, че направения компромис няма съществено да промени получаваните резултати.)

Както и в предходното упражнение се разглежда изпълнението на следните две обобщени групи команди:

• Първа група: В тази група се включват всички аритметични и логически команди, а така също и командите за преход. Те се появяват с вероятност ***p***. С цел уточняване на модела тази група се подразделя на три подгрупи - команди за безусловен преход, команди за условен преход и чисто изчислителни и логически команди. За целите на настоящия модел е прието, че командите за безусловен преход представляват 5% от първата група команди и предизвикват изчистване на конвейера. За командите за условен преход се приема, че представляват 15% и че с вероятност 50% извършват същото като тези за безусловен преход. Третата подгрупа - команди за изчисление и логическа обработка не предизвикват извънредни събития в конвейера. Програмната реализация на този модел дава две възможности - да се отчитат командите за преход и да не се отчитат. Така е възможно да се изследва тяхното влияние върху производителността на конвейера.

• Втора група: В нея са включени командите, извършващи четене/запис от/в паметта, т.е. команди, имащи един операнд, който се прехвърля от паметта във вътрешните регистри на процесора или обратно. При тяхното моделиране се извършва резервиране на кеша за данни. Те се появяват с вероятност **1-*p***.

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на имитационен модел, реализиран чрез програмата **LAB4**, да се проведат следните моделни експерименти:

1. В режим "Демо".

1.1. Какъв е най-големия брой команди съхранявани в опашката и какво събитие в работата на конвейера настъпва след запълване на опашката.
1.2. Всяка команда се идентифицира чрез буква и две числа разделени с точка. Например: Xy.z, където X може да бъде: J – команда за безусловен преход; C – команда за условен преход; M – команда за работа с паметта; R – всички останали команди. Първото число y показва поредния номер на командата. Въпросът е какво показва второто число?

2. В режим моделиране.
2.1. Как влияе изменението на времето за изпълнение на командата (етап 5) върху производителността и средното време за изпълнение?
2.2. Да се определи как се влияе производителността на дългия конвейер при срещане на команди за преход?
2.3. Сравнете получените резултати с тези от предходното упражнение.

КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ

1. Какво налага въвеждането степента "транслация" в конвейера?
2. От какво зависи времето за изпълнение на командата в етап 5?

УПРАЖНЕНИЕ 5

АНАЛИЗ НА ПРОИЗВОДИТЕЛНОСТТА НА ОПЕРАЦИОНЕН КОНВЕЙЕР

ЦЕЛ НА УПРАЖНЕНИЕТО

Да се изследва как се променя производителността на конвейера от дължината обработваемия вектор, броя на степените в конвейера и броя на паралелно работещите конвейери, включени в състава на векторния процесор.

ТЕОРИЯ

Освен при обработката на команди, конвейрът може да се приложи и при обработката на данни. Това се прави във векторния процесор. Така допълнително се увеличава производителността при обработката на регулярни структури от данни, т.е. вектори (матрици).

Векторът е набор от скаларни данни, всички от един и същ тип, съхранявани в паметта. Обикновено векторните елементи са подредени, така че да имат фиксирано адресно нарастване между следващите елементи.

Векторна обработка се среща когато аритметични или логически операции са приложени към векторите. Преобразуването от скаларен към векторен код се нарича ***векторизация***, а съответните компилатори се наричат ***векторизиращи компилатори***.

В скаларните процесори много често данните не са част от командата, а се извличат от паметта чрез посочване на адреси. Декодирането на адреса и извличане на данните от паметта отнема известно време. С цел намаляване на това време, повечето модерни процесори използват техниката, известна като конвейер за команди. Векторните процесори използват тази концепция и я развиват. Вместо само конвейер за командите, те също прилагат конвейер върху данните. Например, ако **A**, **B** и **C** са вектори с **n** елемента, за скаларния процесор сумирането на двата вектора **A** и **B** би изглеждало така:

```
for(i=0, i<n-1, i++)
c[i]=a[i]+b[i];
```

Всяка от тези команди трябва да се декодира и минава през конвейера на скаларния процесор, преди да завърши и така не се получава голямо увеличение на скоростта на изпълнение. Но за векторният процесор тази задача изглежда значително по-различно, а именно:

C=A+B

т.е. за векторният процесор същата задача се решава с една команда. Тази една команда представя многото команди от скаларния процесор; така не само може да се прескочат всичките адресни декодирания, но в крайна сметка има само една команда за декодиране.

Постигнатият ефект от прилагането на конвейер за данните зависи от различни фактори, основните от които са: брой степени в конвейера, брой елементи на обработвания вектор и времето за инициализация на векторната команда.

Единичен конвейер. От теорията е известно, че ако **n** обекта (команди или данни) следват един след друг, то времето за тяхната конвейерна обработка се дава с израза:

$$(L+n-1)*t_k \qquad (5.1)$$

където: t_k е времето между две последователни заемания на една и съща степен на конвейера, т.е. такта на конвейера, а L е броят на степените на конвейера.

При използването на векторна команда за обработка на последователността от данни, т.е. вектор, във формула (5.1) се добавя още едно слагаемо:

$$\mathbf{t_{int}} + (\mathbf{L}+\mathbf{n}-\mathbf{1})*\mathbf{t_k} \qquad (5.2)$$

където: t_{int} е времето за инициализация на векторната конвейера.

От друга страна, времето за обработка на същия вектор с дължина **n**, но на АЛУ без конвейер е $N*t_c$, където t_c е времето за изпълнение на съответната операция. Така увеличаването на производителността на АЛУ с конвейер може да се определи спрямо тази на АЛУ без конвейер като

$$S = \frac{T_1}{T_N} = \frac{n*t_c}{t_{int}+(L+n-1)*t_k} \qquad (5.3)$$

Ако с едно първо приближение предположим, че

$$t_k = t_c / L \qquad (5.4)$$

то за S се получава:

$$S = \frac{L*n}{\sigma+L+n-1} \qquad (5.5)$$

където: $\sigma = \frac{t_{int}}{t_k}$ е времето, в брой тактове на конвейера, за неговата инициализация

От (5.5) е ясно, че конкретната стойност на производителността S зависи от σ, n и L. С увеличаване на **n**, S се стреми към своята пределна стойност - L. От друга страна, с увеличаване на броя степени в конвейера се увеличава производителността. Това е така, защото на всеки конвейерен такт се получава резултат, а времетраенето на такта, макар и непропорционално, намалява с увеличаването на L до една пределна стойност. Същевременно, големи стойности на σ могат да доведат до неефективна обработка на по-късите вектори.

Ефективността на конвейера E се дава с израза:

$$E = \frac{S}{L} = \frac{n}{\sigma+L+n-1} \qquad (5.6)$$

Ако (5.6) се реши спрямо n, може да се определи минималната дължина на вектора – формула (5.7), за която производителността на конвейера се доближава до максималната при приемлива ефективност.

$$n \geq \frac{E*(\sigma+L-1)}{1-E} \qquad (5.7)$$

Макроконвейер. Векторните компютри освен "стандартните" команди за аритметични операции имат и така наречените съставни команди. Тези команди включват в себе си няколко аритметични операции. Конвейерното им изпълнение дава

възможност да се увеличи още повече производителността на АЛУ.

Работата на векторния процесор при изпълнението на съставните команди ще бъде показано на следния пример.

Пример: *Да се изчисли* **f=(a+b)*d**, *където* **a**, **b**, **d**, **f** *са вектори с дължина* **n** *елемента.*

Решаването на задачата ще бъде показано за двата случая:

а) конвейерът не използува съставни команди;

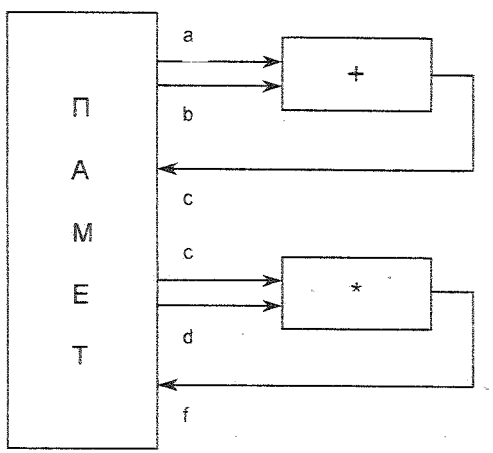
б) конвейерът използува съставни команди.

Случай а) В този случай изчислението на задачата може да се представи като последователност от "стандартни" команди:

$$c = a + b$$
$$f = c*d$$

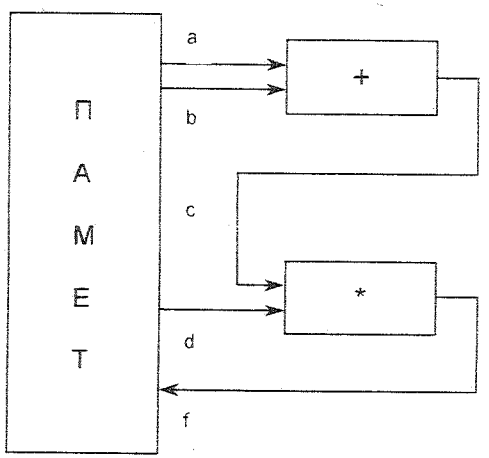
където: ***a***, ***b***, ***c***, ***d*** и ***f*** са вектори с дължина **n**.

Тяхното изпълнение схематично е показано на фиг. 5-1. Най-напред се извличат двата вектора ***a*** и ***b*** от паметта (или от векторните регистри), обработват се от конвейера за сумиране и се записват отново в паметта. След това се стартира конвейера за умножение, който обработва векторите ***c*** и ***d***, като резултатът ***f*** се записва отново в паметта.



Фиг. 5-1. Схема на изпълнение на задачата без използването на съставни команди

Случай б) Във векторният процесор, в чийто набор команди са включени и съставни команди, има команда реализираща израза изцяло, т.е. команда от вида **f=(a+b)*d**. За апаратната поддръжка на съставната команда е необходимо двата конвейера да се свържат последователно - фиг. 5-2. Така се получава един дълъг конвейер (макроконвейер) - неговата дължина е равна на дължината на двата конвейера. Важно е да се подчертае, че съставната команда се третира като едно цяло, а не като съставена от две независими команди.



Фиг. 5-2. Схема на изпълнение на задачата с използването на съставни команди

Ускорението, което се получава при изпълнението на съставната команда спрямо последователното изпълнение на "стандартните" команди – случай а), се дава с израза:

$$S = \frac{\sum_{1}^{k}(L_i + \sigma_i) + k*(n-1)}{\sigma_m + \sum_{1}^{k} L_i + n - 1} \qquad (5.8)$$

където: k е броят на паралелно работещите конвейери;

σ_i е времето за инициализация на всеки конвейер;

σ_m е времето за инициализация на макроконвейера .

Ако се предположи, че са в сила зависимостите:

$$\sigma_1 = \sigma_2 = ... = \sigma_k = \sigma_m = \sigma$$

$$L_1 = L_2 = ... = L_k = L,$$

се получава опростения израз, даващ оценка за производителността на макроконвейера спрямо последователно работещите ***k*** на брой конвейери.

$$S = \frac{k*(L+\sigma+n-1)}{\sigma + k*L + n - 1} \qquad (5.9)$$

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на програма **LAB5** да се проведат следните изследвания:

1.За единичен конвейер.

Да се определи как зависи дължината на обработвания вектор - ***n***, от броя степени на конвейера, т.е. да се определи зависимостта n(L).

Заб. Изследванията да се проведат при една предварително фиксирана стойност на ***E***, намираща се в интервала (0.6.....0.8).

2.За макроконвейер.

Как и с колко се повишава производителността при използуване на макроконвейер? Каква е минималната и максималната стойност на ***S*** ?

Изследванията да се проведат за две различни стойности на σ. Какво е влиянието на σ ?

КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ

1. С увеличаване на ***L*** разликите между t_k, определено от (5.4), и действителното t_k расте. Защо?
2. Колко начина за увеличаване на производителността при обработка на вектори познавате? Кои са те?
3. Можете ли да начертаете структурната схема на векторния процесор и да обясните предназначението на отделните функционални блокове?
4. На колко нива може да се осигури конвейерна обработка във векторния процесор?

5. Може ли да се реализира векторна обработка, ако в процесора не е реализирана конвейерна обработка?
6. Какви препоръки бихте дали на програмиста, разработващ алгоритми за векторна обработка, за да се осигури максимална производителност?
7. Какви са предимствата и недостатъците при използуване на съставни команди?
8. Каква е разликата между съставните команди и дългите команди, използвани в VLIW процесорите?

УПРАЖНЕНИЕ 6

КОМПЮТРИ С SMP И MPP АРХИТЕКТУРИ

ЦЕЛ НА УПРАЖНЕНИЕТО

Да се изследват двата основни типа паралелни архитектури – с обща и с разпределена памет и да се анализира как зависи тяхната производителност от различните фактори.

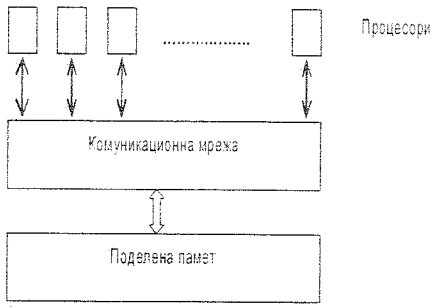
ТЕОРИЯ

В зависимост от отношението на процесорите към паметта се рзличават два основни класа паралелни компютри - силно свързани системи и слабо свързани системи.

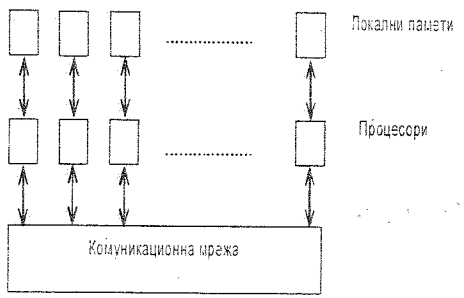
При силно свързаните системи, няколко процесора, чрез комуникационна мрежа директно комуникират с паметта и останалите устройства в системата. При слабо свързаните системи, паралелният компютър се състои от локални компютри, обединени в локална мрежа. Първите системи са познати още като компютри с обща памет – **SMP** (**S**ymmetrical **M**ultiple **P**rocessors), а вторите като компютри с разпределена памет **MPP** (**M**assively **P**arallel **P**rocessing or **M**assively **P**arallel **P**rocessor). На фиг. 6-1 са дадени обобщените структури на двете системи.

При първият клас компютри времената за обмен на информацията между процесорите и средното време за изпълнение на машинните операции са съизмерими, докато във вторият клас времето за обмен на информацията между процесорите е по-голямо от времето за изпълнение на машинните операции.

SMP е един от най-зрелите модели за изграждане на многопроцесорни архитектури. Думата "симетрична" в названието на архитектурата означава, че всеки процесор може да прави всичко, което и всеки от останалите процесори. Понастоящем **SMP** често се разглежда като алтернативно название на компютрите с обща памет, която е поделена между останалите процесори и от тука идва и другата разшифровка на абревиатура **SMP**: **S**hared **M**emory **P**rocessors.



а) Обобщена структура на SMP компютър



б) Обобщена структура на MPP компютър

Фиг.6-1. Обобщени структури на SMP и MPP компютри

В **SMP** компютрите освен няколкото процесора всичко е в един екземпляр: една памет, една операционна система, една подсистема за вход/изход. При този клас компютри всички процесори имат пряк и бърз достъп до паметта. По такъв начин процесорът не просто работи над част от проблема, но той го "вижда" цялостно. В **SMP** архитектурата се опростява както системното, така и приложното програмиране, и това позволява на

една многонишкова операционна система да разпределя задачите си между различните процесори и приложенията да получават толкова памет колкото е необходима. Глобалното поделяне на паметта също опростява синхронизацията на данните. Програмистите на такива системи не се грижат за това къде се намират данните, защото всеки процесор в системата може да се обръща към произволен операнд.

Обаче общата памет е причина и за най-сериозния недостатък на **SMP**: когато се добавят нови процесори, трафика по комуникационната мрежа към паметта нараства бързо, докато достигне точка на насищане, т.е. системата е лошо мащабируема. За решаването на проблема се използват:

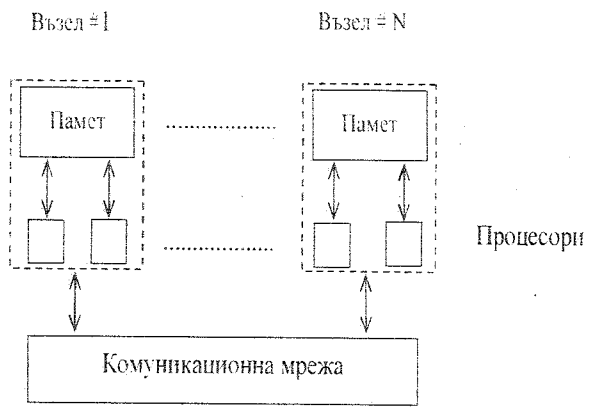
- кеш-памет, включена към всеки процесор;
- подходящо разпределение на данните по модулите памет.

Използването на кеш-памет поражда друг проблем – необходимо е да се осигури кохерентност за данните, използвани от всичките процесори, но получавани от един процесор. Проблемът е известен като съгласуване на данните в паметта (cache coherence problem). Обобщено може да се каже, че тясно място в тези компютри се явява недостатъчната пропускателна способност на каналите за връзка на процесорите към паметта, което и ограничава максималния брой процесори.

Другият клас компютри – **MPP**, е с нестандартна архитектура, предвиждаща използването на разпределена памет. Тези системи съдържат голямо количество процесори – от стотици до няколко хиляди. Всеки процесор има своя собствена памет, към която другите процесори имат непряк и по-бавен достъп. Важно е да се подчертае, че в системата няма обща памет. Така се разрешават конфликтите на процесорите с паметта. За сметка на това е необходимо да се осигури ефективно взаимодействие между процесорите, т.е. да се въведе обмен на данните, разположени в различните памети. Това се реализира с помощта на по-бавните входно-изходни операции на процесора. Практически единствения способ за програмиране на този клас системи е чрез използване на обмен на съобщения. Известни са две технологии – **PVM** (**P**arallel **V**irtual **M**achine) и **MPI** (**M**essage-**P**assing **I**nterface), чието приложение не винаги е просто. Благодарение на големият брой

съвместно работещи процесори е възможно да се реализират системи с масов паралелизъм и да се осигури мащабируемост на архитектурата. Същевременно компютърните архитектури с разпределена памет не позволяват да се изпълняват съществуващите програми с висока скорост, а преработката на програмите отнема време и струва скъпо. Много често е необходимо да се разработят нови паралелни алгоритми.

От изложеното по-горе става ясно, че двата класа имат свои достойнства, които постепенно преминават в техни недостатъци. Може ли да се обединят достойнствата на двата класа компютри? Едно от възможните направления е проектирането на компютри с архитектура **NUMA** (**N**on **U**niform **M**emory **A**ccess).



Фиг.6-2. Обобщена структура на компютър с NUMA архитектура

Ясно е, че всеки възел може да се разглежда като **SMP** компютър. По този начин програма, съхранявана в един модул на паметта, може да се изпълнява от произволен процесор в системата. Единственото различие е скоростта на изпълнение. Всички локални обръщения към паметта, направени от процесор намиращ се в същия възел, се обработват многократно по-бързо отколкото обръщенията направени към останалите модули на паметта и намиращи се в другите възли.

От тази особеност и произтича названието на този клас компютри – компютри с не еднакъв достъп до паметта. В този смисъл, класическите **SMP** компютри са с архитектура **UMA** (**U**niform **M**emory **A**ccess), осигуряващи еднакъв достъп за произволен процесор.

Както и за **SMP** компютрите, и за този клас компютри е характерно несъгласуваност на данните на ниво кеш памет. За решаването на този проблем е разработена специална модификация на **NUMA** архитектурата - **ccNUMA** (**c**ache **c**oherent **NUMA**). За целта са разработени множество протоколи, съгласуващи съдържанието на всички кеш памети и програмиста не се грижи за този проблем.

Резонно е да се постави въпроса "Колко "нееднородна" е архитектурата **NUMA**? Ако обръщението към паметта на другия процесор изисква време от порядъка на 5-10% повече отколкото времето към собствената памет, то една такава система от гледна точка на потребителя ще се "държи" като класическа **SMP** система и практически всички разработени програми за **SMP** компютрите ще се изпълняват и на **NUMA** компютрите. Но за съвременните **NUMA** компютри, разликата във времената за локален и отдалечен достъп е в интервала от 200-700%. При такава разлика в скоростите на достъп е необходимо да се обмисли внимателно правилното разположение на данните по различните модули памет.

В групата компютри с разпределена памет, освен традиционните компютри, посочени по-горе, се включват и кластерните системи. В компютърната литература значението "кластер" се употребява в различен аспект. В частност, "кластерната" технология се използва за повишаване на скоростта и надеждността на сървърите за база данни и Web-сървърите. Тука ние ще обсъждаме само кластерите, ориентирани за решаване на задача от изчислителен характер. В този смисъл кластер означава съвкупност от компютри, обединени в някаква мрежа за решаване на една задача. В качеството на изчислителни възли обикновено се използват достъпни на пазара еднопроцесорни компютри, дву- или четири- процесорни **SMP** сървъри. Всеки възел работи под управлението на свое копие на операционната система. Състава и мощта на всеки възел може да се мени, което дава възможност за създаване на нееднородни системи.

Независимо дали става въпрос за **SMP** или **MPP** архитектури, производителността им зависи от голям брой фактори, част от които са свързани с характера на приложната задача, а други се определят от архитектурните особености.

• Фактори, свързани с решаваната задача:

- Преобладаващ тип аритметични действия – операции с плаваща или фиксирана запетая, броя прости аритметични операции като събиране (изваждане) или сложни операции като умножение (деление).
- Размер на задачата – определя общия брой на аритметичните операции, които трябва да се изпълнят.
- Вътрешен (свойствен) паралелизъм на задачата – определя възможностите за разделяне на изчислителната работа на паралелни сегменти, които се изпълняват едновременно от различни процесори. Задачите с висока степен на вътрешен паралелизъм допускат разделяне (декомпозиция) на по-малки сегменти, докато при други задачи преобладават последователните действия и увеличаването на броя на паралелните сегменти над определена стойност е невъзможно.
- Метод за разделяне на задачата на паралелни сегменти и тяхното разпределение между ресурсите на паралелния компютър – процесори и памет. От него зависи равномерното натоварване на всички процесори, конфликтите при достъпа до общата памет и времето за синхронизация между процесорите.
- Минимален размер на паралелния сегмент (grain size).

• Фактори, свързани с архитектурата на паралелните компютри:

- Брой на процесорите ***N***, състав на множеството инструкции за всеки процесор и брой на скаларните и векторните обработващи устройства, включени в състава на всеки процесор.
- Организация на паметта. Паралелните компютри с обща (глобална) памет предполагат интензивен комуникационен трафик между процесорите и паметта. При архитектури без глобална памет всеки процесор има достъп само до своята локална памет, а достъпът до паметта на другите процесори става чрез предаване на съобщения (message passing). В

зависимост от комуникационните изисквания на конкретната задача се проявяват предимствата на едната от двете архитектури.

- Структура на комуникационната мрежа, която в по-голяма или по-малка степен може да съответства на комуникационните изисквания на приложната задача.

Съществуват различни модели за определяне на производителността на паралелни компютри. В това упражнение са разгледани два модела: първият модел изследва производителността на паралелни компютри с обща памет, а вторият изследва производителността на паралелни компютри с разпределена памет при които процесорите са свързани в хиперкуб. Двата модела предполагат равномерно разпределение на изчислителния товар между процесорите и са достатъчно абстрактни, което позволява да се очертаят основните зависимости между архитектурните особености и техническите показатели на паралелните компютри върху производителността.

А. Модел на паралелен компютър с глобална памет

Един от възможните модели за определяне на производителността на компютъра е да се представи производителността, като функция от параметри, които зависят както от организацията на компютъра, така и от характера на приложната задача. Параметрите са:

• Декомпозиционна функция на изчисленията $D_a(N)$, която се определя от отношението на времето за изчисления в компютър с един процесор $T(1)$ към времето за изчисления в паралелен компютър с N процесора $T(N)$.

$$D_a = \frac{T(1)}{T(N)}$$

$D_a(N)$ определя само присъщия на задачата паралелизъм.

• Декомпозиционна функция на комуникациите $D_c(N)$, която се определя от отношението на броя операции за достъп до данните в еднопроцесорен компютър – $W_c(1)$ към броя операции за достъп до данните в глобалната памет в паралелния компютър $W_c(N)$:

$$D_c(N) = \frac{W_c(1)}{W_c(N)}$$

$D_c(N)$ зависи от архитектурните особености на паралелния компютър. Например при матрични и векторни операции в паралелния компютър с непосредствен достъп на процесорите до глобалната памет съответства декомпозиционна функция $D_c(N) = N$. Когато като операнди се използват само клетки от локалната памет и е необходимо копиране на данните от глобалната в локалната памет и обратно $D_c(N) = 1$.

• Пропускателна способност (bandwidth) на комуникационната мрежа $BW(N)$, която се определя от броя на възможните операции за едновременен достъп до глобалната памет за време t_c (t_c е времето за предаване на число с плаваща запетая между някой от процесорите и глобалната памет).

Пропускателната способност $BW(N)$ зависи както от структурата на комуникационната мрежа, така и от разпределението на данните в паметта. Например за комуникационна мрежа тип **crossbar switch** или йерархични мрежи (**multistage network**) $BW(N)=N$ при условие, че в мрежата няма конфликти при достъп до модулите на глобалната памет. Когато има конфликти, но данните са разположени така, че всеки процесор се обръща към различен модул на глобалната памет, в общия случай $BW(N) = O(\sqrt{N})$. Ако всички процесори се обръщат към един и същи модул на паметта, а също и при архитектури с единствена обща шина $BW(N)=1$.

• Брой на йерархичните нива в комуникационната система $L(N)$.

Така чрез броят на междинните нива в комуникационната мрежа и чрез времето t_c се изразява времето за стартиране на комуникациите t_{st}

$$t_{st} = [L(N)-1]*t_c$$

В случай на многостъпална комуникационна мрежа t_c е времето за предаване на данни от едно стъпало на друго.

За да се оцени ускорението на изчисленията в паралелния компютър Cvetanovic предполага, че задачата се състои от повтарящи се итерации, като при всяка итерация се изпълняват

общо W_a машинни инструкции и W_c операции за достъп до глобалната памет. След отчитане на посочените по-горе фактори той предлага следната формула:

$$S(N) = \frac{T(1)}{T(N)} = min\left\{ D_a(N), \frac{W_a * D_c(N) * BW(N) * t_a}{W_c * N * t_c} \right\} \qquad (6.1)$$

където t_a е средното време за изпълнение на аритметични (логически) операции.

Тази формула разкрива най-общите зависимости между факторите, които определят производителността на паралелни архитектури с глобална памет.

Б. Модел на паралелен компютър с разпределена памет

Моделът на паралелен компютър с разпределена памет се обсъжда на базата на предположението, че процесорите в компютъра са свързани във вид на двоичен хиперкуб.

Двоичният хиперкуб с размерност d се състои от $N=2^d$ процесора, всеки от които свързан непосредствено чрез двупосочни комуникационни канали с d съседни процесора. Всеки от процесорите има достъп до своята локална памет, а посредством предаване на съобщения по комуникационните канали може да се осъществи достъп и до локалната памет на останалите процесори.

За да се отразяват в модела загубите от комуникацията между процесорите на хиперкуба, Amdahl въвежда показател на глобалност G, който се определя от характера на приложната задача и от разпределението на данните между процесорите. Приема се, че когато един процесор се обърне за информация към един от останалите процесори, той ще я намери с вероятност G в един от съседните върхове, с вероятност G^2 – в един от отстоящите на две ребра върхове, с вероятност G^3 – в един от отстоящите на три ребра и т.н. При това предположение средното разстояние – D, на което се предават съобщенията в хиперкуба е:

$$D = \frac{d * G}{1 + G} \qquad (6.2)$$

Ако се приеме, че задачата се състои от итерации, при всяка от които се изпълняват общо W_a инструкции и всеки процесор приема

съобщение с големина W_C, обемът на изчисленията на всяка итерация в хиперкуб с N процесора е:

$$W(N) = W_a + N * W_C * D \qquad (6.3)$$

Вторият член в дясната страна на формула (6.3) отразява увеличението на обема на изчисленията, дължащо се на комуникации при разпределението (декомпозицията) на задачата.

Макар и избраният подход за оценка на времето за комуникация да е твърде абстрактен, той позволява чрез подходящ подбор на параметрите G и W_C да бъде моделирана производителността на хиперкуба за различни класове приложни задачи. Например в идеалния случай, когато необходимите за изчисленията данни са изцяло разположени в локалната памет на процесорите, показателят на глобалност $G = 0$ и вторият член в дясната страна на (6.3) отпада. При условие, че отсъства локализация на данните, показателят приема максималната си стойност $G = 1$ и $D = d/2$, което съответства на комуникация между два случайно избрани процесора.

При условие, че работният товар е разпределен равномерно между всички процесори и времената за изчисления и комуникации са еднакви ($t_a=t_c$), за ускорението на изчисленията може да се напише:

$$S(N) = \frac{T(1)}{T(N)} = \frac{W_a * N}{W(N)} = \frac{N}{1 + \frac{N * W_c * D}{W_a}} \qquad (6.4)$$

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на програмата **LAB6** да се проведат следните експерименти за определяне на производителността на двата типа паралелни компютри.

А) Модел с обща памет:
Изследванията да се проведат за следните гранични случаи:

$D_a = N$, $D_c = \sqrt{N}$, $BW(N) = N$ (crossbar мрежа без конфликти)
$D_a = N$, $D_c = \sqrt{N}$, $BW(N) = \sqrt{N}$ (crossbar мрежа с конфликти)

$D_a = N$, $D_c = \sqrt{N}$, $BW(N) = 1$ (комуникационната мрежа е шина или има обръщения към един и същ модул памет)
$D_a = N$, $D_c = 1$, $BW(N) = N$ (crossbar мрежа без конфликти)
$D_a = N$, $D_c = 1$, $BW(N) = \sqrt{N}$ (crossbar мрежа с конфликти)
$D_a = N$, $D_c = 1$, $BW(N) = 1$ (комуникационната мрежа е шина или има обръщения към един и същ модул памет)

За всички тези случаи W_a = 0.2, 0.8 и t_a/t_c = 5, 10 ,20.

Получените резултати да се анализират и да се дадат отговори на следните въпроси:
1. Как влияе t_a / t_c върху производителността?
2. Как влияе изменението на W_a върху производителността?
3. За кои случаи е подходящо да се използва паралелна система с обща памет.

Б) Модел с разпределена памет:

Да се определят границите на увеличение на ускорението на изчисленията в зависимост от броя процесори в паралелния компютър. Изследванията да отчитат двата случая:

а) груб паралелизъм, т.е. голям размер на паралелните сегменти, при което $W_C < W_a$.
б) фин паралелизъм, т.е. $W_C \geq W_a$.
Препоръчва се да се направят изследвания със следните стойности на W_a:
- W_a = 0.9 (тогава W_c =0.1 и имаме груб паралелизъм);
- W_a = 0.1 (тогава W_c =0.9 и имаме фин паралелизъм);

Получените резултати да се анализират и да се дадат отговори на следните въпроси:
1. За кой от двата случая – груб или фин паралелизъм е по-подходяща архитектура хиперкуб и защо?
2. Как влияе параметърът глобалност върху получаваните резултати?
3. Начертайте графика $S(N)$ в зависимост от G.

КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ

1. Кои са причините за появяването и развитието на паралелните компютри?
2. От какво зависи производителността на паралелните компютри?
3. Какви други свързвания между процесорите познавате освен

хиперкуб?
4. Кои са предимствата (недостатъците) на компютрите с разпределена памет?
5. Каква характеристика на компютъра се отчита с t_a / t_c ?

УПРАЖНЕНИЕ 7

ОПРЕДЕЛЯНЕ НА ПРОИЗВОДИТЕЛНОСТТА НА MPP КОМПЮТРИ ПРИ РЕШАВАНЕ НА ЕДИН КЛАС ЗАДАЧИ

ЦЕЛ НА УПРАЖНЕНИЕТО

Да се определи какво е влиянието на решаваната задача върху производителността на паралелна система, в която ресурсът обща памет отсъства и процесорите комуникират по между си чрез обмен на съобщения.

ТЕОРИЯ

Множеството процесори са свързани по различен начин, и се управляват от един централен компютър (най-често това е стандартен компютър). Изчислителният процес протича по следния начин:

Централният компютър изпраща данни и команди в паметта на подчинените процесори. След това всички процесори започват да работят едновременно и автономно. Това е фазата на паралелната работа. След нейното завършване в общия случай трябва да се обмени информация между подчинените процесори, а също така и между тях и централния компютър. Това е фазата на последователната работа. Фазите на паралелна и последователна работа се редуват. Ясно е, че както потока от данни, така и потока от команди се обработват асинхронно.

Основен критерий за оценка на производителността на един паралелен компютър е времето за решаване на една задача – ***T(P)*** (***P*** е броят на процесорите). В най-общия случай то може да се представи като сума от следните времена:

$$T(P) = T_c(P) + W(P)$$

където: $T_c(P)$ е времето за аритметични и логически операции и за изпълнението на инструкциите за управление;

W(P) е времето, изразходвано за комуникация, синхронизация и разпределение на задачата между всички процесори.

Прието е като оценка на производителността на паралелната обработка да се използва ускорението на изчисленията – ***S(P)***. То се дефинира като:

$$S(P) = \frac{T(1)}{T(P)} = \frac{T(1)}{T_c(P) + W(P)}$$

или

$$S(P) = \frac{1}{\frac{T_c(P)}{T(1)} + \frac{W(P)}{T(1)}}$$

Този израз може да се опрости като се предположи, че $\frac{T_c(P)}{T(1)} = \frac{1}{P}$, т.е. приложната задача се разделя само на паралелни сегменти, чийто брой е кратен на **P**. В някои случаи е необходимо да се отчита влиянието на крайната разделяемост на приложната задача. То се проявява, когато паралелните сегменти не са с еднакъв обем данни за обработка. За целта се въвежда коефициента σ, отчитащ дали натоварването на процесорите е балансирано (σ = 1) или не е (σ > 1). Така за ***S(P)*** се получава:

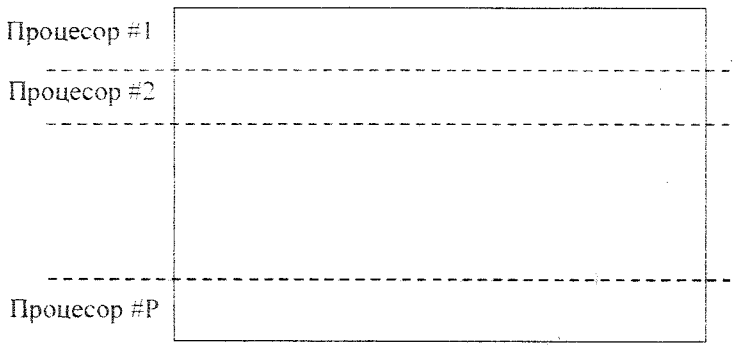
$$S(P) = \frac{1}{\frac{\sigma}{P} + \frac{W(P)}{T(1)}} \qquad (7.1)$$

Коефициентът σ зависи в крайна сметка от решавания проблем и метода (алгоритъма) за решаването му. ***W(P)*** зависи както от ***P***, така и от модела на изчисление и топологията на комуникационната мрежа. Определянето на тези коефициенти ще се конкретизират по-долу на базата на решаване на един клас задачи, свързани с линейната алгебра и по-точно събиране (изваждане) на матрици, умножение на матрици, транспониране на матрица, намиране на обратна матрица, решаване на системи линейни уравнения и др.

Метод за решаване.

Паралелното решаване на тези задачи може да бъде реализирано по различен начин, като тук ще бъде разгледан само един от тях. За решаването на посочените по-горе задачи се прилага метода на декомпозиция върху обработваните матрици. Същността му е следната: Матрицата **A(n,n)** (за простота ще се предполага, че

матрицата е квадратна), се разделя на хоризонтални (вертикални) слоеве (подматрици), чийто брой е равен на броя на подчинените процесори – фиг. 7-1.



Фиг.7-1. Хоризонтално разделяне на матрицата A.

Всеки слой съдържа ***n/P*** реда на матрицата ***A***. Той се съхранява в паметта на съответния процесор. Така всеки процесор обработва само "своята" част от матрицата ***A***.

Определяне на σ .

Ако ***P*** се нанася цяло число пъти в ***n***, това означава, че всички процесори обработват еднакъв брой редове от матрицата. В този случай **σ = 1** и се казва, че процесорите са натоварени равномерно.

Ако ***P*** не се нанася цяло число пъти в ***n***, то тогава един или няколко процесора ще обработват повече редове от останалите процесори. В този случай **σ > 1** и се казва, че процесорите не са натоварени равномерно и следва да се очаква намаляване на производителността. В този случай σ се определя по формулата:

$$\sigma = \frac{\left\lceil \frac{n}{P} \right\rceil}{\left\lfloor \frac{n}{P} \right\rfloor}$$

където $\lceil \bullet \rceil$ означава най-малкото цяло число по-голямо от частното, а символът $\lfloor \bullet \rfloor$ - най-голямото цяло число по-малко от частното.

Определяне на W(P).

Най-голям проблем представлява определянето на конкретни стойности на функцията на загубите - **W(P)**. Това е така, защото тя зависи от различни фактори като: топология на свързване на процесорите, вид на приложната задача, организация на изчислителния процес. Във функцията на загубите се включва и времето за разпределение на задачата по процесорите, зареждането на данните и други системни издръжки.

Ако се предполага, че са изпълнени следните условия:

а) няма препокриване на входно/изходните с изчислителните операции;

б) времето за декомпозиция е пренебрежимо малко;

в) паметта на всеки процесор е достатъчно голяма за да съхранява всички данни, без да се налага допълнителен обмен между външната памет и локалните памети, то е в сила равенството: $W(P)=t_{comm}(P)$, където $t_{comm}(P)$ е време за предаване на данните между процесорите.

Поради изключително сложния характер за определяне на $t_{comm}(P)$, ще оценим $t_{comm}(P)$ за най-лошия случай – при последователно предаване на данни между процесорите, т.е. в даден момент се предават данни само между една двойка процесори. На практика така се елиминира до голяма степен влиянието на топологията на свързване на процесорите. Така стойността на $S(P)$ получена от (7.1) ще дава една оценка ограничена отдолу за реалната стойност на $S(P)$. Оценката за $S(P)$, ограничена отгоре, е известна от теоретичната постановка на задачата за определяне на производителността и тя е $S(P)=P$.

Съобразявайки се с посоченото по-горе, в табл.7-1. са дадени формулите за определяне на $t_{comm}(P)$ за разглежданите задачи при обработване на матрици.

Таблица 7-1.

№	Задача	$t_{comm}(P)$
1.	Събиране (изваждане) на матрици	$1.5n^2(P-1))t_c$
2.	Умножение на матрици	$2n^2(P-1)t_c$
3.	Транспониране на матрица	$n^2(P-1)t_c$
4.	Намиране на обратна матрица	$3n^2(P-1)t_c$

В табл.7-1 променливата t_c е времето за предаване на един елемент от матрицата **A** между два съседни процесора. Например, времето за предаване на един 32 битов аргумент между два процесора (транспютъра) T805/30 е 2.4 μs.

На края, за да се определи ***S(P)*** по формула (7.1), е необходимо да се знае ***T(1)***. Тези стойности се определят чрез измерване с тестови програми, като резултатите са поместени в табл.7-2. (Резултатите са валидни за T805/30, 32 битова аритметика и език за програмиране ANSI C Toolset на INMOS.)

Таблица 7-2.

№	Задача	T(1) [μs]
1.	Събиране (изваждане) на матрици	$4.173n^2$
2.	Умножение на матрици	$4.264n^3$
3.	Транспониране на матрица	$3.661n^2$
4.	Намиране на обратна матрица	$8.963n^3$

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на програмата **LAB7** да се изследва производителността на паралелната система с цел да се определи:

1. За всяка задача, решавана на паралелен компютър с конкретен брой процесори да се определи критичният й размер. (Под критичен размер на задачата се разбира онази минимална размерност- ***n*** на матрицата **A**, за която има смисъл използването на паралелна обработка, т.е. ***S>)1***.

2. Да се определи каква е зависимостта на ***S(n)*** при ***P=const*** за всяка задача, за която има смисъл въвеждането на паралелна обработка. Коя е възможната най-висока производителност?

3. Кои са причините за да получават локални минимуму и максимуми в графиката ***S(n)*** ?

4. Да се определи каква е зависимостта на ***S(P)*** при ***n=const*** за всяка задача, за която има смисъл въвеждането на паралелна обработка.

КОНТРОЛНИ ВЪПРОСИ ЗАДАЧИ

1. Защо е необходимо да се осигури равномерно натоварване на процесорите ?
2. Как бихте определили ***T(1)***, ако няма възможност да се използват тестови програми за определянето му ?
3. Какво се отчита с параметърът σ ?

УПРАЖНЕНИЕ 8

ОПРЕДЕЛЯНЕ НА ОСНОВНИТЕ ПАРАМЕТРИ НА НЯКОИ СТАТИЧНИ КОМУНИКАЦИОННИ МРЕЖИ

ЦЕЛ НА УПРАЖНЕНИЕТО

Да се изследват основните параметри на някои от статичните комуникационните мрежи и да се определи как зависят те от броя на процесорите.

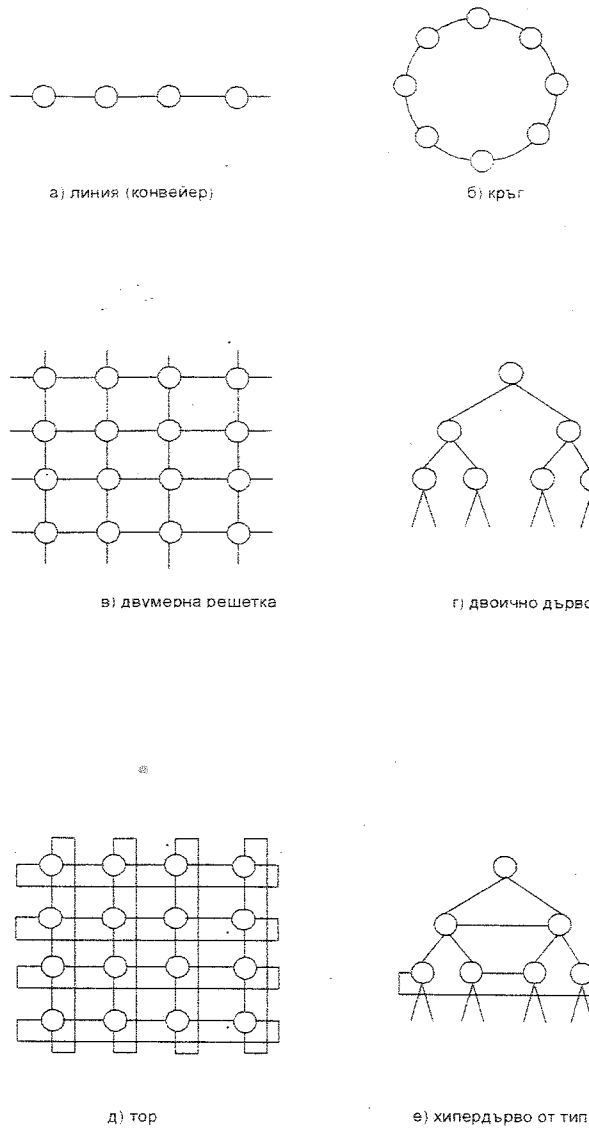
ТЕОРИЯ

Една от особеностите на паралелните компютри спрямо последователните е необходимостта от обмен на информацията между процесорите, за да се получи крайното решение на проблема. За целта се използува комуникационна мрежа (**КМ**), от чийто параметри до голяма степен зависи общата производителност на компютъра. Недостатъчното бързодействие и/или ограничените комбинационни възможности на **КМ** могат съществено да снижат очаквания ефект от повишаване на бързодействието.

В общия случай **КМ** представлява ***NxM*** многополюсник, входните ***N*** полюса са присъединени към предавателите, а изходните ***M*** полюса са присъединени към приемниците. **КМ** се изграждат от комуникационни елементи (**КЕ**) и линии за връзка (**ЛВ**). С помощта на **КЕ** се свързват **ЛВ** и така се осигурява път за предаване на информацията от предавателите към приемниците. По-нататък, за целите на настоящето упражнение, се разглежда само случая когато ***N=M***.

Всяка **КМ** може да бъде описана като граф, чийто върхове съответстват на **КЕ**, а дъгите съответстват на **ЛВ**. Този граф е прието да се нарича мрежова топология. Тя е ключов фактор в определянето на възможностите на **КМ** от гледна точка на архитектурата на паралелния компютър.

КМ биват статични и динамични. В това упражнение ще се спрем само на статичните **КМ**. На фиг.8-1 са дадени някои примери за статични комуникационни мрежи.



Фиг.8-1. Примери на статични комуникационни мрежи

Статичната комуникационна мрежа осигурява непосредствена връзка само между строго определени компоненти, най-често компютри. Връзките с останалите компоненти се определят индиректно. Като правило статичните **КМ** работят с комутация на пакети, като ролята на **КЕ** изпълняват самите процесори. Статичните **КМ** намират основно приложение при изграждането на кластъри от компютри.

За сравнителна оценка на различните статични **КМ** се използуват следните основни параметри:

N - брой на компютрите в мрежата.
D - диаметър на мрежата. Дефинира се като максималното разстояние между произволна двойка върхове в пътя, свързващ двата върха. В случая под разстояние се разбира минималния брой на дъгите, свързващ двата върха.
P - средно разстояние. Определя се като средно разстояние на пътищата от всеки връх до всички останали върхове.
T - средна плътност на трафика. Определя се като брой съобщения за единица време върху една линия за връзка, т.е.

$$T = \frac{N * P}{\text{общ брой на линиите за връзка в мрежата}}$$

Освен тези основни параметри за всяка статична мрежа може да се посочи наличието (отсъствието) на алтернативни пътища за връзка между две произволни двойки процесори. Естествено, наличието на повече пътища за връзка (разбира се те могат да бъдат с различни разстояния), ще създаде условия за по висока отказоустойчивост на паралелния компютър. Пример за такива мрежи са решетка, хиперкуб, тор, кръг с хорди и др., но не и конвейер (линия) или кръг. Друг параметър е мрежата да бъде изградена от еднотипни елементи, т.е. от всеки връх (процесор) да излизат един и същ брой дъги (**ЛВ**), при това този брой да не зависи от броят на процесорите. Така се гарантира лесна мащабируемост на паралелния компютър. Примери за такива мрежи са решетка, кръг, кръг с хорди, тор, двоично дърво, **k**-ичния хиперкуб (k≥2), но не и двоичния хиперкуб.

В настоящето упражнение за определянето на основните параметри на всяка мрежа, последната се представя като граф и чрез известен алгоритъм за намиране на пътищата в граф, напр. на Форд или Дейкстра, се определят **D**, **P** и **T**.

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на програма LAB8 да се получат резултати за D, P и T за различните топологии на свързване и при различен брой процесори - **N**. В резултат на изследванията да се определи:

1. Как се променя ***D***, ***P*** и ***T*** с увеличаване на ***N***? Каква е зависимостта на **D(N)** за всяка една от топологиите?

2. Подредете по възходящ ред топологиите в зависимост от параметрите ***D***, ***P*** и ***T***, т.е. да се определи коя топология е по-добра и по кой параметър.

3. Коя топология бихте препоръчали да се използва като най-добра?

КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ

1. Какво представлява в най-общия случай една **КМ**? Къде е мястото ѝ в общата структура на компютъра?
2. Какви **КМ** познавате?
3. Какви са характеристиките на всяка **КМ**?
4. По какъв начин се определя пътя между предавателя и приемника.
5. С какви параметри се оценяват статичните мрежи?

УПРАЖНЕНИЕ 9

АНАЛИЗИРАНЕ НА ПРОИЗВОДИТЕЛНОСТТА НА НЯКОИ ДИНАМИЧНИ КОМУНИКАЦИОННИ МРЕЖИ

ЦЕЛ НА УПРАЖНЕНИЕТО

Целта на упражнението е да се изследва производителността на два представителя от класа на динамичните мрежи, а именно на **cross-bar** мрежа (матричен превключвател) и **delta** мрежа.

ТЕОРИЯ

Динамичните **КМ** дават възможност за промяна на комуникационните връзки посредством реконфигурация на активни **КЕ**. Те осигуряват директна връзка между компонентите на паралелния компютър и са еднакво подходящи както за схемна комутация, така и за комутация на пакети. Като правило те се изграждат от няколко стъпала, като по този начин се опростява структурата на всеки **КЕ**, но за сметка на това се увеличава времето са предаване на данни.

Параметрите по които се оценяват динамичните **КМ** са:

• Сегментиране: Това е възможността на **КМ** да се разделя на независими подмрежи с различни размерности. Всяка подмрежа трябва да има всички възможности за връзка на цялата мрежа от същия тип и размери. Следователно със сегментирането на мрежата, паралелния компютър може да се разглежда като множество от паралелно работещи паралелни компютри и така по друг начин да се реши задачата за защита на данните (програмите), а така също и за отказоустойчивостта на компютъра.

•Ширина на лентата на пропускане. Този параметър се дефинира като очакван брой заявки приети за единица време. Тъй като системната шина не може да осигури широка лента на пропускане, а матричния превключвател е твърде скъп, то е интересно да се знае как варира лентата на пропускане за различните **КМ**. За целта се използуват различни аналитични методи, а за по-точни

резултати (и разбира се при наличието на мощни компютри) и на подходящи симулационни програми.

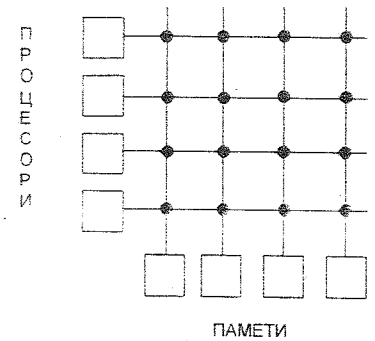
В това упражнение се анализират ***N*** **x** ***M*** мрежи с матрично превключване (**cross-bar**) и делта мрежи за свързване на ***N*** процесора с ***M*** модула памет. Оценките и за двата вида мрежи се базират основно на работа на Janek H. Patel и са получени при едни и същи предположения, за да има база за сравнение. Тези предположения са:

1. Всеки процесор генерира случайно и независимо заявки към паметта. Заявките са равномерно разпределени върху всички модули на паметта.
2. В началото на всеки цикъл всеки процесор генерира заявка с вероятност ***p***. По такъв начин ***p*** е също така средният брой заявки, генерирани за един цикъл от всеки процесор.
3. Заявките, които са блокирани, т.е не са приети, се игнорират.
4. Заявките, подадени в следващия цикъл, са независими от блокираните заявки.

Последното предположение е прието за опростяване на анализа. Въпреки че този модел не отчита отхвърлените заявки, все пак той е полезен, тъй като може да бъде решен точно и дава долната граница за очакваната ширина на пропускане. На практика отхвърлените заявки трябва да бъдат представени в следващия цикъл или да бъдат записани в модула, където възниква конфликтът. Това означава, че предположението за независимост на заявките не се изпълнява. По-късно последното предположение ще бъде отслабено, за да се подобри моделът. Нещо повече, публикуваните резултати от моделирането на процеса на обмяна на информация показват, че вероятността за приемане се намалява незначително при отпадане на третото предположение. По такъв начин резултатите от анализа са достатъчно надеждни и са добра мярка за сравнение на различни мрежи.

<u>CROSS-BAR мрежа</u>.

Примерна топология на cross-bar мрежа (матричен превключвател) е дадена на фиг.9-1.



Фиг.9-1. Топология на cross-bar мрежа

В една пълна мрежа от този тип две заявки са в конфликт тогава и само тогава, когато се отнасят към един и същи модул от паметта. Ето защо по същество се анализират конфликти в паметта, а не конфликти в мрежата. Нека ***g(i)*** да е вероятността всички ***i*** –заявки да пристигнат в един цикъл. Тогава

$$g(i) = P_i * p^i * (1-p)^{n-i} \qquad (9.1)$$

където P_i е биномният коефициент.

Нека освен това E_i да бъде очакваният брой заявки, приети от ***N*** x ***M*** cross-bar мрежата за един цикъл, при условие, че ***i*** заявки са пристигнали в този цикъл. Известно е, че броят на начините, по които ***i*** случайни заявки могат да се разпределят между ***M*** различни модула на паметта е M^i. Ако някой от модулите не е избран, то броят на комбинациите, по които ***i*** заявки могат да се разпределят по останалите ***(M-1)*** модула е $(M-1)^i$. Следователно $M^i-(M-1)^i$ е броят на разпределенията на заявките, при които даден модул получава винаги заявка. Тогава вероятността за неговото използване е

$$\frac{M^i-(M-1)^i}{M^i}$$

От това следва, че очакваният брой приети заявки, при условие, че са постъпили ***i*** заявки, е:

$$E(i) = \frac{M*(M^i-(M-1)^i)}{M^i} \qquad (9.2)$$

По такъв начин очакваната широчина на лентата на пропускане *B(N,M)*, т.е средният брой на заявките, приети за един цикъл, е:

$$B(N,M) = \sum_0^n E(i)*q(i),$$

което се опростява до

$$B(N,M) = M - M*(1-\frac{p}{M})^N \qquad (9.3)$$

След това се определя вероятността, че произволна заявка ще бъде приета

$$p_A = \frac{B(N,M)}{p*N} = \frac{M*(1-(1-\frac{p}{M})^N)}{p*N} \qquad (9.4)$$

Уравнение (9.4) е изведено при допускането на независимост на заявките. В действителност отхвърлената заявка не се игнорира, а отново се подава в следващия цикъл, като по такъв начин се увеличава скоростта на подаване на заявките. Разглежданият модел може да се допълни като се предположи, че отново появилите се заявки са равномерно разпределени по адресите на паметта.

Вероятността ***p*** на постъпване на заявките трябва да се определи по-точно като вероятност, която осигурява безконфликтен достъп. Освен това заявки към паметта се отправят и през всеки загубен цикъл. Следователно има една друга динамична вероятност на постъпване на заявките, отбелязана с ***p'***, която е по-голяма от статичната ***p*** поради конфликтите в паметта. Динамичната вероятност ***p'*** може да бъде получена от графа на Марков и се дава с израза:

$$p' = \frac{p}{p+p_A*(1-p)} \qquad (9.5)$$

В този случай уравнение 9.4 придобива вида:

$$p_A = \frac{M * (1 - (1 - \frac{p'}{M})^N)}{p'N}. \qquad (9.6)$$

Уравнения 9.5 и 9.6 определят итеративен процес, по който може да се изчисли p_A, за дадени ***M***, ***N*** и ***p*** .

Оттук следва, че p_A представлява мярка за изгубените цикли. По-малката стойност на p_A показва по-голям брой изгубени цикли и обратно – по-голямата стойност на p_A показва по-малък брой изгубени цикли. Средният брой изгубени цикли ***L*** на една заявка може да се определи, ако се вземе предвид, че една заявка, която е отхвърлена, чака ***L*** цикъла.

$$L = \frac{1 - p_A}{p_A} \qquad (9.7)$$

DELTA мрежа.

Мрежата е с размери a^k x b^k и е построена от матрични модули ***axb***. По такъв начин a^k процесора са свързани с b^k модули памет (**k** е броят на стъпалата в мрежата). На фиг.9-2 е показана delta мрежа при ***a = b = 2*** и **k = *3***.

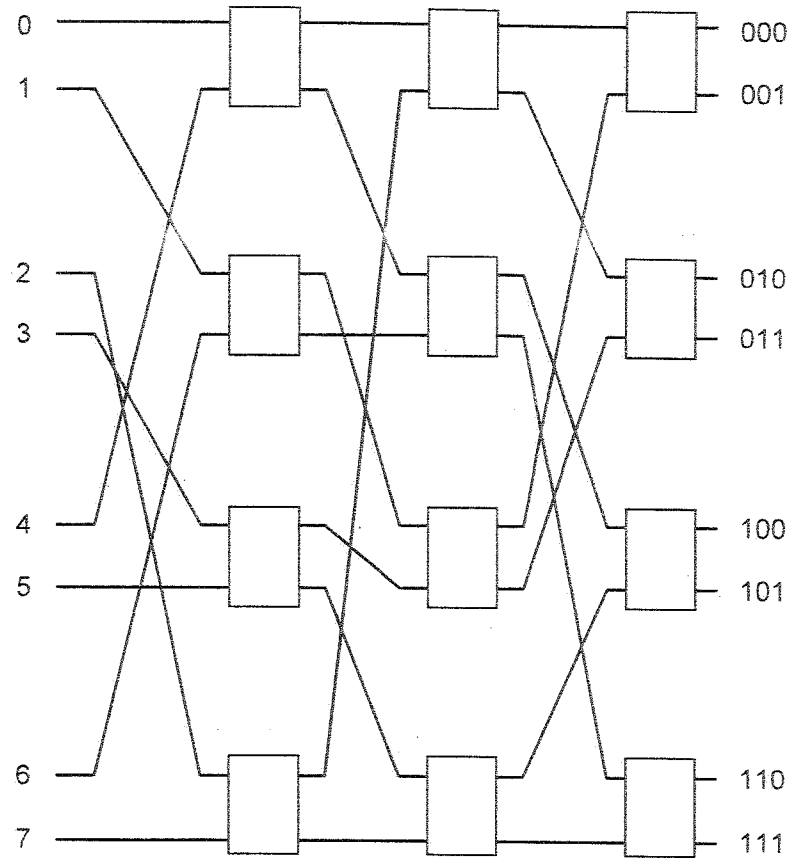
Ако се използва резултатът, получен в (9.3) за матричния комутатор, лесно може да се направи анализ и за **delta** мрежата.

При зададена вероятност ***p*** за възникване на заявки на всеки вход на матричния превключвател очакваният брой заявки, които минават през него за единица време, се получава като в (9.3) като се замести **N = *a*** и **M = *b***, което дава:

$$B(N, M) = b - b * (1 - \frac{p}{b})^a$$

Като се раздели горният израз на броят на изходните линии – ***b*** на модула ***a*** x ***b*** , се получава вероятността за възникване на заявки на всяка от изходните *b* линии:

$$1 - (1 - \frac{p}{b})^a \qquad (9.8)$$



Фиг. 9-2. Пример за delta мрежа

По такъв начин за всяко стъпало на delta мрежата изходната скорост на появяване на заявки p_{out} представлява функция на входната скорост p_{in}:

$$p_{out} = 1 - (1 - \frac{p_{in}}{b})^a \qquad (9.9)$$

Тъй като изходната вероятност на едно стъпало̂ е входна вероятност за следващото стъпало, може рекурсивно да се изчисли изходната вероятност за който и да е етап, започвайки от първия. В частност изходната вероятност на крайното стъпало **k**

определя широчината на пропускане на **delta** мрежата, т.е. броят на заявките, преминали за един цикъл.

Нека p_i е вероятността на подаване на заявки на изходна линия на стъпало i. Тогава широчината на лентата на пропускане $B(a^k, b^k)$ за **delta** мрежата $a^k \times b^k$ при зададено p е:

$$B(a^k, b^k) = b^k * p_k \qquad (9.10)$$

където: $p_i = 1 - (1 - \frac{p_{i-1}}{b})^a$ и $p_0=p$ и $i=1,2,...k$..

Тъй като няма решение в явен вид на уравнение (9.10), широчината на лентата на пропускане на **delta** мрежата не може директно да се сравни с широчината на лентата на пропускане на **cross-bar** мрежата. Сравнението е възможно да се направи само по графичен път.

Вероятността една заявка да бъде приета е:

$$P_A = \frac{b^k * p_k}{a^k * p} \qquad (9.11)$$

Средният брой изгубени цикли се определя по (9.7).

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на програмата **LAB9** да се изследват характеристиките на двата типа мрежи. И за двете мрежи p се променя по програмен път в границите *0.1, 0.2 ...1.0.*

CROSS-BAR мрежа
Изследванията да се проведат за двата случая:

1. Мрежата е квадратна, т.е. $N = M$ = const (N = 2, 4, 8, 16, 32, 64).
2. Мрежата е правоъгълна.

За да бъдат коректни сравненията между квадратната и правоъгълната мрежи е необходимо **N** и **M** така да бъдат подбрани, че и в двата случая броят на използваните комутиращи

елементи да бъде едно и също число. Очевидно е, че при правоъгълната мрежа е възможно да се отделят следните два подслучая: а) **N>M**; б) **N<M**, които следва да се разгледат самостоятелно.

Получените резултати да се анализират, за да се дадат отговори на следните въпроси:

1. Как се променят B и L за трите вида мрежи в зависимост от p ?
2. Коя мрежа – квадратна или правоъгълна има по-голямо **B** и **L** и кога?

Delta мрежа

1. Сравнете две **delta** мрежи по основните параметри B и L, които имат еднакъв брой входове (изходи), но различен брой стъпала.

2. Сравнете трите различни **delta** мрежи, за които е в сила:
   а) N=M;
   б) N>M;
   в) N<M.
3. Да се сравнят получените резултати с тези за cross-bar мрежата. Коя мрежа има предимство и кога?

КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ

1. Колко основни вида комуникационни мрежи познавате? Какви са техните предимства?
2. С какви параметри се оценяват динамияните мрежи?
3. Какви други динамични мрежи познавате освен разгледаните тук?

УПРАЖНЕНИЕ 10

ИЗСЛЕДВАНЕ ВЛИЯНИЕТО НА КЕШ ПАМЕТТА ВЪРХУ ВРЕМЕТО ЗА ДОСТЪП ДО ДАННИТЕ

ЦЕЛ НА УПРАЖНЕНИЕТО

Да се изследва какво е увеличението на скоростта на достъп до информацията за компютър, притежаващ кеш памет спрямо компютър само с основната памет и от какви фактори зависи. Също така, ние се интересуваме и от средното време за достъп и ефективността на системата кеш памет - основна памет.

ТЕОРИЯ

Работа на кеш-паметта

Кеш паметта е ключа за осигуряване на производителност при съвременните процесори. Типично в една програма около 25% от инструкциите се отнасят до паметта, така че времето за достъп до паметта се явява критичен фактор в изпълнението на програмата. Чрез ефективно редуциране времето за достъп до паметта, кеш паметта позволява в модерните процесори да се изпълняват повече от една инструкция за цикъл.

Работата на кеш паметта се базира на локалността на обръщенията. Всички програми показват локалност на обръщенията. Това се оказва универсално свойство на програмите – независимо дали са комерсиални, научни, игри и т.н. Кеш паметта експлоатира това свойство, подобрявайки времето за достъп и намалява цената на достъп до главната памет. Има два типа локалност

а) локалност по време;
б) локалност по пространство (специална локалност).

Локалност по време - Веднъж направено обръщение към дадена позиция (клетка) от паметта има голяма вероятност, че тя ще бъде потърсена отново в недалечно бъдеще.

Примери: Най-простият пример за локалност по време е изпълнението на цикъл: веднъж въведен (включен) цикълът,

всички команди в цикъла ще се посочват отново, вероятно много пъти, преди цикълът да завърши. Обобщено извикването на подпрограми, функции и прекъсванията по време също проявяват такова свойство.

Много типове данни показват локалност по време: в някаква точка на програмата съществува тенденция за обръщение към "горещи" данни, които програмата използва или променя многократно, преди да премине към друг блок от данни. Някои примери са:

- броячи;
- преглед на таблици;
- натрупване на променливи;
- стекови променливи.

Локалност по пространство - Когато се адресира клетка от паметта, много вероятно е, че съседните клетки ще бъдат скоро достъпни.

Примери: Ясно е, че потокът от инструкции ще проявява значителна специална локалност. В отсъствие на преходи следващата команда да бъде изпълнена е едно непосредствено, пряко следствие на текущата.

Данните също показват значителна специална локалност – напр. когато матрица или низ са достъпни, обработката започва от началото на матрицата до края, последователно.

Локалността по време включва локалността по пространство, но не и обратно.

Модел на работата на кеш-паметта

При разработването на модела, даващ отговор на поставените по-горе цели, е удачно да се въведат следните означения:

- p - вероятност, че търсения обект (данни или команди) се намира в кеш паметта;
- t_c - време за достъп до кеш паметта;
- t_m - време за достъп до основната памет.

Като се използуват приведените по-горе определения, то средното време за достъп t, както то изглежда за процесора, се явява сума

от средното време за достъп до кеш паметта и до основната памет, т.е.

$$t = pt_c+(1-p)t_m \qquad (10.1)$$

Така скоростта за достъп до информацията за компютър само с основна памет е $1/t_m$, а за компютър, в който има и кеш памет, е $1/(pt_c+(1-p)t_m)$. Тогава увеличението на скоростта за достъп при компютър с кеш памет се получава:

$$S = \frac{t_m}{t} = \frac{a}{p+(1-p)a} \qquad (10.2)$$

където: $a=t_m/t_c$ и показва колко пъти е по-бърза кеш паметта от основната памет.

От тези величини ***p*** се явява критична и представлява най-голяма грижа за проектанта на кеш паметта. Максималното значение е равно на 1, но то не е достижимо поради:

- невъзможността да се осигурят винаги обръщения към последователни адреси от паметта;
- голямо различие между обема на основната памет и кеш паметта, достигаща често два и повече порядъка.

Значението на ***p*** не може да бъде равно и на 0 вследствие на:

- повторно използуване на част от информацията.

В действителност параметъра ***p*** никога не се явява константа. Той се изменя динамично за различните програми, изпълнявани на един и същ компютър. Освен това структурата на кеш паметта също влияе върху ***p***. Тези особености не се отчитат в разгледания аналитичен модел, което се явява едно негово ограничение. Независимо от това получените чрез него резултати показват едни от основните зависимости при използуването на КЕШ ПАМЕТТА.

Ефективността на използуваната кеш паметта се дефинира като:

$$E=S/a \qquad (10.3)$$

и показва каква част от общото бързодействие на системата кеш памет – основната памет се определя от бързодействието на кеш паметта.

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на програма **LAB10** да се получат резултати за ***t*** при различни стойности на t_m и t_c, зададени от ръководителят на упражнението. Да се анализират резултатите по отношение на:

1. Какво е влиянието на ***p*** върху **S** и **E** и за какви негови стойности е рационално използуването на кеш памет?

2. Какво е влияето на коефициента ***a*** върху **S** и **E** и при какви негови стойности се получава по-голяма ефективност от използуването на кеш паметта?

КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ

1. Какво е мястото на кеш паметта в общата структура на паметта в компютъра?
2. Какво представлява локалността на обръщение по време? Дайте примери.
3. Какво представлява локалността на обръщение по простанство? Дайте примери.
4. Какви структури на кеш паметта познавате?
5. Какво се разбира под понятието „ефективност" на използването на кеш паметта? Какво означава E=0 и E=1?
6. Защо вероятността за намиране на информация в кеш паметта в една реална система не може да бъде нито 0 нито 1?
7. В реалният свят, от какво зависи вероятността ***p*** за намиране на търсената информация в кеш паметта?

УПРАЖНЕНИЕ 11

ИЗСЛЕДВАНЕ ВЛИЯНИЕТО НА КОНФЛИКТИТЕ ПРИ ОБРЪЩЕНИЕ КЪМ ПАМЕТТА ВЪРХУ ЕФЕКТИВНОСТТА Й

ЦЕЛ НА УПРАЖНЕНИЕТО

При компютри с разслоена памет, ако процесорът се обръща към паметта, без да е завършило обслужването на предходната заявка, генерирана от същия процесор (или от друг процесор), възниква конфликт, определящ престой на процесора (процесорите). Целта на упражнението е да се анализира влиянието на тези конфликти върху ефективността на системата оперативна памет – централен процесор.

ТЕОРИЯ

При хоризонталната организация на паметта, целта не е да се намали времето за достъп, както при вертикалната организация, а да се увеличи броят на достъпите за единица време. Резултатът в крайна сметка е един и същ – по-бърз достъп до информацията. Тази организация засяга само основната памет и се прилага когато въвеждането на кеш памет е невъзможно или нецелесъобразно, напр. във векторните компютри. Разделянето на кеш паметта от първо ниво на кеш за команди и кеш за данни също може да разглежда като разновидност на тази организация.

Съществуват различни начини за хоризонтална организация, най-простият от който се състои във физическо разбиване на паметта на две половини, достъпът до който може да бъде осъществен едновременно. Поместването на всички команди в едната половина, а данните в другата, увеличава средната скорост на достъп към паметта. Това по същество е Харвардската архитектура.

По-нататъшното развитие на тази идея води до използуването на няколко модула памети, свързани към независими процесори с помощта на високоскоростна комуникационна мрежа, така че всички модули на паметта са еднакво достъпни за всички процесори.

Хоризонтална организация на паметта може да бъде изградена по два начина:

- пакетна обработка на множество достъпи към паметта;
- конвейерна обработка на множество достъпи към паметта.

Предложеният по-долу аналитичен модел се явява обобщен и се базира на теорията на марковските процеси.

Нека компютъра съдържа **N** процесора и **M** модула памет. Последователните адреси на паметта са разположени в различни, но последователни модули. Системното време се измерва в тактове, а пълният цикъл на паметта заема t такта.

Предполага се, че в началото на всеки такт всеки процесор, който се намира в състояние на чакане, може да се обръща към паметта с вероятност p. Обръщението към произволен модул е равновероятно. Ако модулът, към който процесора се обръща, не е завършил обработката на предходната заявка, то времето на чакане за освобождаването на модула е равномерно разпределено в интервала от 0 до t. Вероятността, че модулът е зает, не зависи от времето и за всички модули е една постоянна величина (това означава, че се предполага стационарен процес).

Ефективността на системата - E, определена като отношение на очаквания брой обръщения към паметта върху сумата на тази величина и времето за престой на процесора в състояние на чакане, е:

$$E = \frac{pr}{pr + (1 - r)}$$

където: $r = \dfrac{2}{1 + \sqrt{1 + \dfrac{2Np^2t(t+1)}{M}}}$

Така окончателно за E се получава:

$$E = \frac{2p}{2p - 1 + \sqrt{1 + \frac{2Np^2t(t+1)}{M}}} \qquad (11.1)$$

Формула (11.1) не дава много добро приближение за статистическа оценка на функционирането на голяма част от реалните компютри. Основните причини за това са:

• Вероятността за обръщение към паметта на всеки такт в действителност зависи от предисторията. Обикновено процесорът определено време се обръща към паметта, след което той прекратява за известен период обръщенията.

• Последователните обръщения се осъществяват или към последователните модули, или с постоянно изместване, така че обръщенията към различните модули не е равновероятно.

• Не се отчита, че няколко процесора чакат обслужване от даден модул.

Независимо от това, формула (11.1) съдържа в явен вид полезна информация за взаимната връзка между такива параметри на системата, като броя на процесорите, броя на модулите памет и продължителността на цикъла на паметта.

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на програма **LAB11** да се анализира ефективността на взаимодействието на системата оперативна памет – централен процесор.

1. Как зависи ***E*** от времето за достъп до паметта – ***t*** ? В какъв диапазон са тези промени? Кога те са най силно изразени?

2. Как зависи ***E*** от броят модули памет и броят процесори? За целта да се направят изследвания за следните случаи:

• M > N;
• M = N;
• M < N.

Кога ефективността на взаимодействието между модулите памет и процесорите е най-голямо? Има ли случаи когато за различните отношения между **M** и **N** се получават едни и същи (или много близки) стойности за ***E*** ?

КОНТРОЛНИ ВЪПРОСИ И ЗАДАЧИ

1. Каква е разликата между хоризонталната и вертикалната организация на паметта?
2. Колко подхода за хоризонтална организация на паметта познавате?
3. Какви са недостатъците на разслоената памет?
4. Кога възникват конфликти при обръщенията към разслоена памет?
5. Използува ли се хоризонтална организация на паметта в едно процесорните системи? Кога и защо?
6. Синтезирайте схема, формираща адреси за разслоена памет.

УПРАЖНЕНИЕ 12

АНАЛИЗ НА ПРОИЗВОДИТЕЛНОСТТА НА ДИСКОВАТА ПАМЕТ

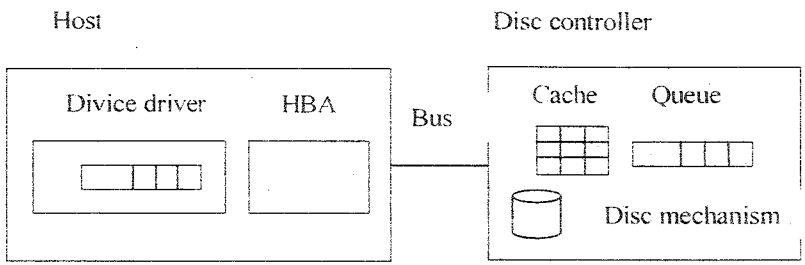
ЦЕЛ НА УПРАЖНЕНИЕТО

Да се изследва какво е влиянието на всеки един от параметрите на дисковата памет върху общата й производителност и да се посочат границите, в които трябва да се намира всеки един от тях.

ТЕОРИЯ

Моделиране работата на един твърд диск изисква да се познава много добре неговото функциониране, затова ще дадем едно кратко изложение на неговото функциониране.

Пътят на входно/изходните операции при твърдите дискове се състои от драйвер на диска, шинен адаптер, шина, контролер и дисков механизъм – фиг.12-1.



Фиг.12-1. Опростен модел на дисковата памет

Драйверът на диска е част от входно/изходната подсистема на операционната система. Той управлява специфичните за устройството взаимодействия, като по този начин изолира тези детайли от останалия системен софтуер. Драйверите често съдържат опашки, които преподреждат заявките с цел подобряване на времето за отговор. Драйверът е свързан с шинния адаптер, който предвижва данните към/от шината.

Шинният адаптер (**HBA - Host Bus Adapter**) е свързан с дисковия контролер чрез интерфейсния протокол. Контролерът обработва заявките, управлява шината и изпълнява кеширане и планиране на заявките към дисковия механизъм.

Дисковете съдържат механизъм и контролер. Механизмът е съставен от записващи компоненти (въртящи се дискове и глави, осигуряващи достъпа до тях) и позициониращи компоненти (рамо, което премества главите на правилната позиция заедно със следяща система, която го държи на място). Дисковият контролер съдържа микропроцесор, буферна памет и интерфейс на шината. Контролерът управлява запомнящото устройство и трансфера на данни към/от механизма и реализира съответствието между постъпващите логически адреси и физическите дискови сектори, които съхраняват информацията.

Предложеният модел симулира работата на един или повече дискове, присъединени към обща SCSI шина. Моделът дава възможност да се задават параметрите както на диска – табл.12-1, така и на модела – табл.12-2.

Табл.12-1. Параметри на диска

Sector Size	Размерът на секторите на диска; в байтове
Sector Per Track	Броят на секторите на пътечка
Track Per Cylinder	Броят на пътечките на цилиндър
Cylinders	Броят на цилиндрите на диска
Cylinder Skew	Определя времето за позициониране на главата при превключване на цилиндрите. Задава се в сектори
Track Skew	Определя времето за превключване между главите. Задава се в сектори
Disc PRM	Обороти на диска за минута; в [rpm]
Bus Garb Time	Времето за установяване на контрол върху шината (ако е свободна); в [μs]
Bus Speed	Пропускателна способност на шината;в [MB/s]
Cache Size	Размерът на кеша; в [KB]

Табл.12-2. Параметри на модела:

Number Of Discs	Броят на дисковете
Number Of Requests	Общият брой на заявките към дисковете
Maximum Time Between Requests	Максималното време между заявките; в [ms]
Maximum Request Size	Максимален размер на заявката към дисковете; в [KB]

При задаване на времето между заявките, трябва да се има предвид, че е направено допускане за равномерно разпределение на заявките между дисковете. Това означава, че при еднакво зададено време и различен брой дискове, средното време между заявките на един диск ще е по-голямо при по-голям брой дискове.

Като всеки модел и този е развит при следните предположения и допускания:

• времето за търсене се моделира, като се използва функция от вида:

$$SeekTime(dis) = \begin{cases} 0 & dis = 0 \\ a + b\sqrt{d} & 0 < dis \le e \\ c + d * dis & dis > 0 \end{cases}$$

• Няма зониране на диска, т.е. има само една зона.

• Не се използва агресивен оптимистичен подход за позициониране на главата преди операция за четене.

• Приема се, че плочите на диска са съвършени и се използват за съхранение на данни всички цилиндри. Поради това логическите и физическите адреси на секторите съвпадат.

• Времето, необходимо на контролера да обработи заявката, се приема за постоянно и от порядъка на 1 ms.

• Размерът на кеш реда е равен на размера на сектора.

• Няма сегментиране на кеш паметта на диска, т.е. не могат да се поддържат няколко потока от данни.

• Заместващият алгоритъм в кеш паметта е от типа **FIFO**.

• Реализира се read-ahead, като прехвърлянето на сектори в кеш паметта може да продължи до неговото запълване, ако няма други заявки към диска. Приема се, че read-ahead може да се прекъсне по средата на трансфера на сектор.

• Всички заявки за запис се записват в кеш паметта. Политиката за запис е immediate write behind.

• Големината на самата заявка се приема за 50 байта, а "done massage" – на 1 байт.

• Ако пристигне заявка за четене или запис на сектори, които не могат да се съберат в кеш паметта, тя се разбива на по-малки заявки с големина равна на тази на големината на кеш паметта.

МОДЕЛНИ ИЗСЛЕДВАНИЯ

С помощта на програмата-модел **LAB12** да се изследва производителността на дисковата система. Изследванията да се проведат с цел определяне на:

1. Как се влияят средното време за достъп и броят на прехвърлените сектори с увеличаването на броя дискове?

Експериментите да се проведат с 1, 2, 3 и 4 диска като се отчитат двата случая:

а) без промяна на времето между заявките;

б) с промяна (пропорционална) на времето между заявките.

2. Промяната на скоростта на въртене на диска върху кой параметър оказва най-съществено влияние.

Експериментите да се направят с 1 диск, като всички параметри остават непроменени с изключение на оборотите на диска.

3. Какво е влиянието на пропускателната способност на шината върху параметрите на дисковата система.

Да не се забравя, че промяната на скоростта на работа на шината е свързана с изменение на параметъра "време за установяване контрол върху шината".

КОНТРОЛНИ ВЪПРОСИ ЗАДАЧИ

1. От какво зависи обслужването на една заявка, отправена към диска?
2. Какво представлява времето за изчакване? От какво зависи то?
3. Как се дефинира "пропускателната способност" на един диск?

4. Какво е характерно за обработката на транзакции от входно-изходната подсистема?
5. Защо времето за обработката на транзакции не се влияе съществено от намаляване на времето за предаване?
6. Защо при предаване на данни от/към дисковата система, свързани с обработката на научни изследвания, всяко усъвършенстване в работата на дисковата система оказва влияние?
7. Какво представляват дисковете матрици с излишък? Какви са техните предимства и недостатъци?

СПИСЪК

на най-често срещаните съкращения
(по азбучен ред)

А. Български съкращения

АЛУ	Аритметично – логическо устройство
КЕ	Комуникационен Елемент
КМ	Комуникационна Мрежа
ЛВ	Линия за Връзка
ОП	Оперативна Памет
ПЕ	Процесорен Елемент

Б. Английски съкращения

CISC	**C**omplex **I**nstruction **S**et **C**omputer/**C**ode
EU	**E**xecution **U**nit
FIFO	**F**irst **I**n **F**irst **O**ut
IFU	**I**nstruction **F**etch **U**nit
MFLOPS	**M**illion **FL**oating point **O**perations **P**er **S**econd
MIPS	**M**illion **I**nstructions **P**er **S**econd
MPP	**M**assively **P**arallel **P**rocessing or **M**assively **P**arallel **P**rocessor
MPI	**M**essage **P**assing **I**nterface
NUMA	**N**on **U**niform **M**emory **A**ccess
RISC	**R**educed **I**nstruction **S**et **C**omputer/**C**ode
VPM	**V**irtual **P**arallel **M**achine
SMP	**S**ymmetrical **M**ultiple **P**rocessors or **S**hared **M**emory **P**rocessors
UMA	**U**niform **M**emory **A**ccess

ЛИТЕРАТУРА

1. Цв. Таслаков.
Компютърни архитектури. www.sc.tu-varna.bg

2. Справочник по компютърна архитектура.
http://foldoc.org/contents/architecture.html

3. В.В.Воеводин, Вл.В.Воеводин.
Параллельные вычисления. БХВ-Петербург, 2002.

4. Цв.Таслаков.
Компютърни архитектури. Печатна база при ТУ-Варна, 2001

5. B.Wilkinson, M. Allen
Parallel Programing. Techniques and Applications Using Networked Workstations and Parallel Computers. Prentice Hall, 1999.

6. Ts. Tasslakov
Analysis of Multiprocessor Structures Built with Transputers. Second Workshop on Parallel Computing and Transputer Applications. Varna 93, ACMBULL Newsletter Vol3, No1, 1993

7. Амамия М., Танака.
Архитектура ЭВМ и исскуственный интелект., Мир, Москва, 1993.

8. Я.А.Илиев, Л.Х.Милков, К.Л.Боянов, Н.В.Синягина.
Аналитични модели за оценка на производителността на паралелни ЕИМ. Автоматика и изчислителна техника. No 1, 1991, стр. 16-23

9. Ст. Марков.
Изчислителни системи с висока производителност. София, Техника, 1990.

10. Мотоока Т., Томита С., Танака Х., Сайто Т., Уэхара
Компьютеры на СБИС. Мир, Москва, 1988.

11. Р. Хокни, К. Джесхоуп.
Параллельные ЭВМ. Москва, Радио и связь, 1986.

12. П.М.Коуги
Архитектура конвейерных ЭВМ. Москва, Радио и связь, 1985.

13. С. О. Степанян
Коммуникационные сети в многопроцесорных ЭВМ. "Автоматика и вьчислительная техника", 1987, #3, с.31-43.

14. Б. Байцер.
Микроанализ производительности вычислительных систем. Москва, Радио и связь, 1983.

15. Д. Ферари
Оценка производительности вычислительных систем. Москва, Мир, 1981.

16. Janak H. Patel
Performance of Processor-Memory Interconnectios for Multiproccessors. IEEE trans. on Comp. Vol. C-30, No 10, October 1981.

17. T. Feng.
A Servy of Interconnection Networks., Computer, 1981, vol.14,#12, p.12-27.

КОМПЮТЪРНИ АРХИТЕКТУРИ
ръководство за лабораторни упражнения

Автор: © Цветан Таслаков
© Милен Ангелов

ISBN 978-954-20-0484-4

Пор. № 27/2010 Формат 16x60x84
Тираж 300 Печ. коли 5. 75
Протокол № 2/02.11.2009 г. Изд. коли 5.36
Учебното пособие е освободено от ДДС
по чл. 41.т.3 от ЗДДС.

Университетско издателство при ТУ-Варна