**1. Модели машинна архитектура и обработка. Класификация и метрика. Мултипроцесори: UMA, NUMA, COMA. Векторни и потокови машини и систолични матрици. Мултикомпютри.**

**Класове компютърни архитектури.** Комп. арх. дефинира компонентите и организацията на една система. Фон Ноймановата арх. се използва при възли и мрежи при некласическа организация (систолични, потокови, логически и редукционни модели и невронни мрежи). Ще разгледаме класификацията на Майкъл Флин за архитектури по управление на потока инструкции и покота данни (операнди) – SISD **(фиг.1.1.)**, SIMD, MISD, MIMD. SISD е класическа архитектура. Останалите се използват от машини за паралелна обработка. SIMD се използва за векторна обработка, фина грануларност. MISD – за конвейрна обработка (обработващи фази върху вектор) – систолични масиви, MIMD – обикновено с локална и глобална памет; за средна и едра грануларност. Класификацията на паралелните архитектури е технологично-ориентирана: мултипроцесори, мултикомпютри, потокови машини, матрични процесори, конвейерни векторни процесори и систолични матрици – частично съответствие с класовете на Флин. **HW/SW (хардуерен/софтуерен) паралелизъм**. Паралелизмът представлява максималният брой инструкции на 1 програма, които може да се изпълняват при обработката на тази програма. За паралелно изпълнение на програми е небходима едновременно апаратна и програмна поддръжка. Апаратния (хардуерния) паралелизъм се обуславя се от архитектурата и ресурсите, които са баланс между производителността и цената. Характеризират се с пикова производителност и средно натоварване. Той задава зависимостта по ресурси. Програмен паралелизъм се обуславя от зависимостта по данни и по управление. Реализира се като: 1) паралелизъм по управление - конвейризация, мултиплициране на функционални възли. Обслужва се паралелно, прозрачно за програмиста.2) паралелизъм по данни - типичен за SIMD и MIMD. **Метрика: ускорение и ефективност.** Ускорението (speed up) е S(n)=T(1)/T(n), а ефективността – E(n) = S(n)/n (нормирана стойност на ускорението); n е броят процеси, ако арх. е фон Нойманова, то n=1, ако имаме повече от 1 процесор: n>1; Редно е S(n) > 1. Най-добрият случай е – включвайки n процеса да намалим времето n пъти **(фиг. 1.2.)**: Линията HW е на ъгъл 45 градуса; SW се определя от контекста на проблема и е независима от n. При стойности над HW – аномалии, под нея – ограничения са наложени. Пр. ако имаме масив от n елемнта – макс. паралелизъм е n (осигуряваме асинхронна операция върху всяка негова клетка). Графиката винаги започва от т.(1,1). Нашето ускорение (кривата) се стреми към SW. Ако сменяме параметрите, то ще получим фамилия от криви. Обикновено имаме нужда от синхронизация в края или началото на програмата. **Делене на обработката: грануларност. (фиг. 1.3.)** Важни са свойствата linearity (линейност – стремеж ускорението да бъде плътно до линията HW – виж горе) и scalability (мащабируемост). Още по-важно е грануларността – размерът на използваните процеси и начина, по който те разпределят проблема. Нивата на грануларност са 5. Фината грануларност (coarse) е на ниво компилатор – при цикли, вектори – прилагане на еднотипна операция върху няколко елемента. Средната грануларност ена ниво подпрограми и процедури, редът, в който ще се изпълняват клоновете на програмата. Очакваме, че при фина гранулация ще нараства броят на процесите, но това увеличава и използваните ресурси. **SIMD(Single instruction, Multiple Data).** Използва се най-вече при машини за векторна обработка**.(фиг. 1.4.)**. Обобщеният модел включва контролно устройство и еднотипни обработващи модули с достъп към обща памет. Програмно-апаратна зависимост на паралелизма /ускорението – пример за изпълнение на програма на SIMD машина **(фиг.1.5А., фиг. 1.5Б.).**Процесорните елементи изпълняват операциите във формат битове или думи. локалната памет за данните може да бъде разпределена, обща или йерархична (със свързваща мрежа). Особености: 1)опростена архитектура спрямо MIMD поради общото контролно устройство (за дешифриране и зареждане на инструкциите) и съответно поддържане само на едно копие от кода за инструкции;2) скаларните операции (включително контролната логика) се изпълняват от контролното устройство – евентуално конкурентно на паралелната обработка на данни в обработващите устройства;3) имплицитна синхронизация между отделните обработващи устройства (при MIMD – експлицитна). Примери – фамилия Connection Machine на Thinking Machine Co. При SIMD се достига най-голям паралелизъм – в някои изчислителни центрове броят на изпълнителните елементи е над 10000. **MISD (Multiple Instruction – Single Data)** **(фиг.1.6. )** Това е архитектурния принцип на всички конвейри – вкл. на процесорния конвейер – обработката се разделя на последователни фази; обработката на следващата инструкция (при най-фина грануларност) или на следващия процес започва веднага щом предходния процес освободи първата фаза. Закъснението при отделните фази (stages) трябва да е равно, не трябва да има бавни. Прилагат се и функционални (или циклични) конвейри например с фазите: четене на инструкциите от обща памет, зареждане в обработващото устройство с евентуално буфериране,обработка, пренос на резултата към общата памет (буфериране), запис в общата памет. Същуствеват няколко нива на конвейеризация: инструкционно, субсистемно (обикн. при аритметична обработка – нелинейни конвейри с фази add, mul, div, sort…) и системно ниво (процеси, също и програмна организация) на конвейризация. **Систолични матрици (Systolic Arrays) -** представляват модификация на MISD на субсистемно ниво, специализирана архитектура за определени алгоритми – с многодименсионни конвейри т.е. фиксирана мрежа от обработващи устройства. Имат ограничено приложение – ЦОС (цифрова обработка на сигнали – DSP), обработка на образи и др. Имат опростени процесорни елементи и комутационна съобщителна мрежа с ограничен набор шаблони. управлението е по инструкции (control flow – не data flow) но програмирането е като при потоковите архитектури. Архитектурата включва обработващ масив (с комутатор) и управляващ модул, който настройва масива, предава данните и извлича резултатите (+ контролен възел – хост) **(фиг. 1.7.).** Производителността се понижава значително при интензивен вход/изход. Има и топологични шаблони: 1) систолични вектори – по същество конвейри; 2) двудименсионни масиви –обикновено регулярни с коеф. на съседство най-често 4 или 6 **(фиг. 1.8., фиг.1.9.)** Тенденцията е към елемeнти за фина грануларност – на инструкционно ниво – снабдени с няколко високоскоростни дуплексни серийни канали (броя на които определя валентността – коеф. на съседство). Пример: iWrap серия на Интел и университета Carnegie-Mellon – процесорната клетка се състои от: iWrap компонент с изчислителен и комуникационен агент и страницирана памет с директен интерфейс към компонента. Пример: умножение на матрици в двумерен систоличен масив с коеф. на съседство 6 **(фиг. 1.10.)MIMD (Multiple Instruction – Multiple Data) (фиг. 1.11.)** Това е архитектурния принцип на всички мултипроцесори и мултикомпютри. Процесорите са автономни и могат да изпълняват различни програми (вкл. локално копие на ОС!). Имат общ ресурс с разпределен конкурентен достъп – памет или комуникационна среда. Организация: 1)автономни (локална памет) - общо адресно пространство (общодостъпна памет);2) магистрални – комутационни. Характеризират се с универсални, отказоустойчиви, по-едра грануларност. Обикновено се изграждат с масови процесори (вместо специализирани процесорни елементи с ограничени функции). Наличието на автономна локална памет ги разделя на:1) системи с обща памет; синоними: мултипроцесори | [shared-memory |tightly-coupled] systems | Global-Memory MIMD, GM-MIMD | Uniform Memory Access System – UMA;2) системи с обмен на съобщения; синоними: мултикомпютри, [distributedmemory | loosely-coupled] systems | Local-Memory MIMD, LM-MIMD | Non-Uniform Memory Access System – NUMA (поради наличието на локална и отдалечена памет). Разполагат с глобално и локално адресно пространство; виртуалната памет поддържа глобално адресно пространство на страниците (не на ниво думи), което се управлява от разпределена ОС (РОС) за мултипроцесори и хомогенните мултикомпютри. При мултикомпютри общата виртуална памет се поддържа и с обмен на съобщения. Хетерогенните мутликомпютри използват мрежови ОС (МОС), при които нивото на

достъп е разпределена файлова система (напр. базирана на DNS) с ползване на примитиви от типа rlogin, rcp... **Мултикомпютри (разпределени машини).** Използват NUMA(Non-Uniform Memory Access System). Характеризират се с разпределената обща памет (distributed shared memory DSM): програмната имплементация на обща памет в система с автономни възли (и адресни пространства). Има виртуално общо адресно пространство от страници (не думи) – 4/8 kB – (което позволява програмиране за мултикомпютъра като за виртуален уникомпютър). При отсъствие на страница от локалната памет възниква вътрешно прекъсване (memory trap) и зареждане на страницата в локалната от отдалечената памет. Възможно е репликиране на страници само за четене (read only). **(фиг. 1.12)** – 1,2,3 са компютри – процесите в компютрите са свързани помежду си с обща памет. Обаче, ако 1 иска да достъпи страница №10, тя трябва да е read only – така се имитира общо адресно пространство. Ако страницата е и за запис, се прилагат различни мерки за поддържане на свързаност. Принципът е приложим и при системи с обмен на съобщения – Message passing distributed systems. **Архитектура с обща памет**

**(мултипроцесори) – UMA -** (uniformly shared memory access) - еднакъв достъп на процесорите -

силносвързани системи. Характеризират се с: 1) обща шина - разширение от унипроцесинг към мултипроцесинг; недостатък – трябва да итерираме достъпа; 2) комутируема матрица (crossbar switch) **(фиг. 1.13)** – свързваме някоя обща памет с определен процесор – рядко разпространение;3) многоканални мрежи **(фиг. 1.14).** Паралелните интерфейси са бързи на много къси разстояния **(фиг. 1.15.)** – пистите, по които тече токът; ако увеличим големината на пистите, скоростта пада. Следователно трябва да променим формата им, с цел да се различат отделните изпращачи на сигнали. При кодиране на сигнала **(фиг. 1.16.)** – ако искаме да изпратим 0, забавяме честотата (това се използва за моделите, които се слагат на 32 или 64b магистрала. Видове синоними: симетричен (централизиран В/И) и асиметричен (специализиран процесор за В/И) мултипроцесинг - обикновено хомогенни системи. **NUMA и CUMA** - NUMA(non-uniformly shared memory access) – йерархия на общата памет - локални, глобални и/или клъстерни памети **(фиг. 1.17.)** и CUMA (cache only shared memory access) - паметта е лакална (cache) но йерархията и позволява част от нея (“директория”) да се адресира отдалечено **(фиг. 1.18.).** И двата модела се използват при мултикомпютрите. **Потокови архитектури (Data Flow)** При класическите фон Нойманови архитектури (вкл. модификациите по Флин) програмата е последователност от инструкции, която се изпълнява от контролно устройство – control flow. При потоковите архитектури операциите се изпълняват веднага при наличие на операндите (и наличие на операционен ресурс) – контрола се осъщесвява чрез планиране на операндите т.е. данните; концептуално всички инструкции с готови операнди могат да се изпълнят паралелно (на практика конкурентно). Програмите за потокови архитектури се представят с потокови графи (обикн. с текстов синтаксис) – възлите представят операции, а дъгите – информационните връзки на операндите; нивото на паралелизъм обикновено е инструкционно **(фиг. 1.19.)** – X = (A+B)\*(C-D): 1)Add A, B; 2)Store T1; 3)Sub C, D; 4)Store T2; 5) Mul T1, T2; 6) Store X. (A,B,C,D – имена на променливи, които компилаторът транслира до относителни адреси; когато програмата се зареди – тези адреси са вече абсолютни. Процесорът работи с относителни адреси). **Статични потокови архитектури.** При тях програмният (потоковия) граф е фиксиран. За изпълнение на повече от една програма се използват различни варианти на зареждането на данните, които се генерират на етапа компилация. Този модел не поддържа процедури, рекурсия и обработка на масиви. Организация – **фиг. 1.20.** .Съществуват статични потоци с реконфигурация - логическите връзки между процесорните елементи се установяват на етапа зареждане на програмата: топологията на връзките се решава от компилатора и след зареждане на програма остава фиксирана при изпълнението; Особености: 1) физическите канали съществуват, но са комутират; 2) броя алоцирани (заредени) процесори обикновено е по-малък от инсталираните процесори поради ограничения в комутацията – логическата връзка между процесорите е дърво, не всички процесори в листата на което се използват;3) пример – MIT Data Flow Machine – клетките памет съответстват на информацията във възлите на потоковия граф – т.е. инструкционните блокове (tokens) – когато блока е комплектован с операнди, той се предава като операционен пакет към елемент за обработка; пакета с резултата се връща в клетъчната памет**(фиг. 1.21.)**. **Динамични потокови архитектури.** Базират се на логически канали между процесорите, които могат да се реконфигурират по време на изпълнение подобно на система с обмен на съобщения – с маркирани блокове (tagged tokens). Дъгите в потоковия граф могат да съдържат повече от един блок едновременно (но с различни марки!). Операциите се извършват когато възела получи блокове (с еднакви марки) на всичките си входящи дъги. Циклични итерации могат да бъдат изпълнявани паралелно: за целта всяка итерация се представя като отделен субграф като маркировката се разширява с номера на итерацията **(фиг. 1.22. )** (само при информационна независимост на итерациите!). Пример – Manchester Data Flow Machine MDM: цикличен конвейер, в който блоковете циркулират и се управляват от ключов модул. Компонентите са: 1) Блоков буфер (token queue) – за съхраняване на междинни резултати (ако се произвеждат по-бързо отколкото е последващата им обработка) – капацитет 32К блока и производитилност 2.5 МБлока/Сек; 2) Комплементираща памет (matching store) – за комплементиране на блоковете с еднакви марки – процеса е апаратен и поддържа до 1.25 МБлока; 3) Памет инструкции (instruction store) – n-торките (обикновено 2ки) операнди-блокове се пакетират с инструкции и адрес (етикет) на резултата и се предават за изпълнение. **Съпоставка на компютърните архитектури.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип | Принцип на действие | Интерфейс | Приложимост | Сложност | Ефективност |
| SIMD | *спонтанен* | *директен* | *средна* | *висока* | *висока* |
| MIMD | *сложна*  *абстракция* | *най-сложна*  *организация* | *висока*  *(универсални)* | *висока* | *средна* |
| MISD | *спонтанен* | *директен* | *ниска* | *ниска* | *висока* |
| Систолични | *сложна*  *абстракция* | *директен* | *ниска* | *средна* | *висока* |
| Потокови | *сложна*  *абстракция* | *сложна*  *организация* | *висока* | *висока* | *висока* |

**Мрежи за връзка.** Осъществяват комуникациите между процесорните възли при всички видове мултипроцесори и мултикомпютри – статични и динамични (базират се на [каскади от] комутируми блокове - ключове). Топологии на свързване: пълен граф, линия и пръстен, двудименсинна циклична и ациклична мрежа, хиперкуб (n-куб), двоично дърво, shuffle exchange. При мултипроцесорите комуникационния метод е чрез обща шина – централизирано се свързват с паметта. При мултикомпютрите – суперканал – имаме арбитраж на заявките – broadcasting (един предава към всички в своята група). Логическата топология е \* - т.е. от всеки към всеки. Има отлагане на заявката, достъпът не е веднагически. Мрежите, в които няма broadcast, използват някаква топология – разпределяне: това е централизиран подход, който усложнява схемата (напр. P2P). Топологията дефинира релация на съседство – връзката между съседи е пряка, а между несъседи – непряка. За да се поддържат топологии, се използват каскадни комутатори – превключват серийните канали, свързващи двойки възли **(фиг. 1.23.)**. **Характеристика на мрежите за връзка.** 1) разстаяние *dij -*диаметър на мрежата *D* = max{*dij* , за всяка двойка(*i*, *j*)} –изисква по-голям бройканали между възлите,респ. валентност;

2) валентност на възлите(degree)3) сечение (bisectionwidth) *S* = min{AllLinks(X, Y):||X| - |Y|| ≤ 1}; 4) разширяемост.

|  |  |  |
| --- | --- | --- |
| Топология | Брой възли | Валентност |
| Линия и пръстен | d | 2 |
| Двоично дърво | 2^d - 1 | 3 |
| Shuffle exchange | 2^d | 3 |
| Двудеменсионна мрежа | d^2 | 4 |
| Хиперкуб | 2^d | d |
| Пълен граф | N | N-1 |