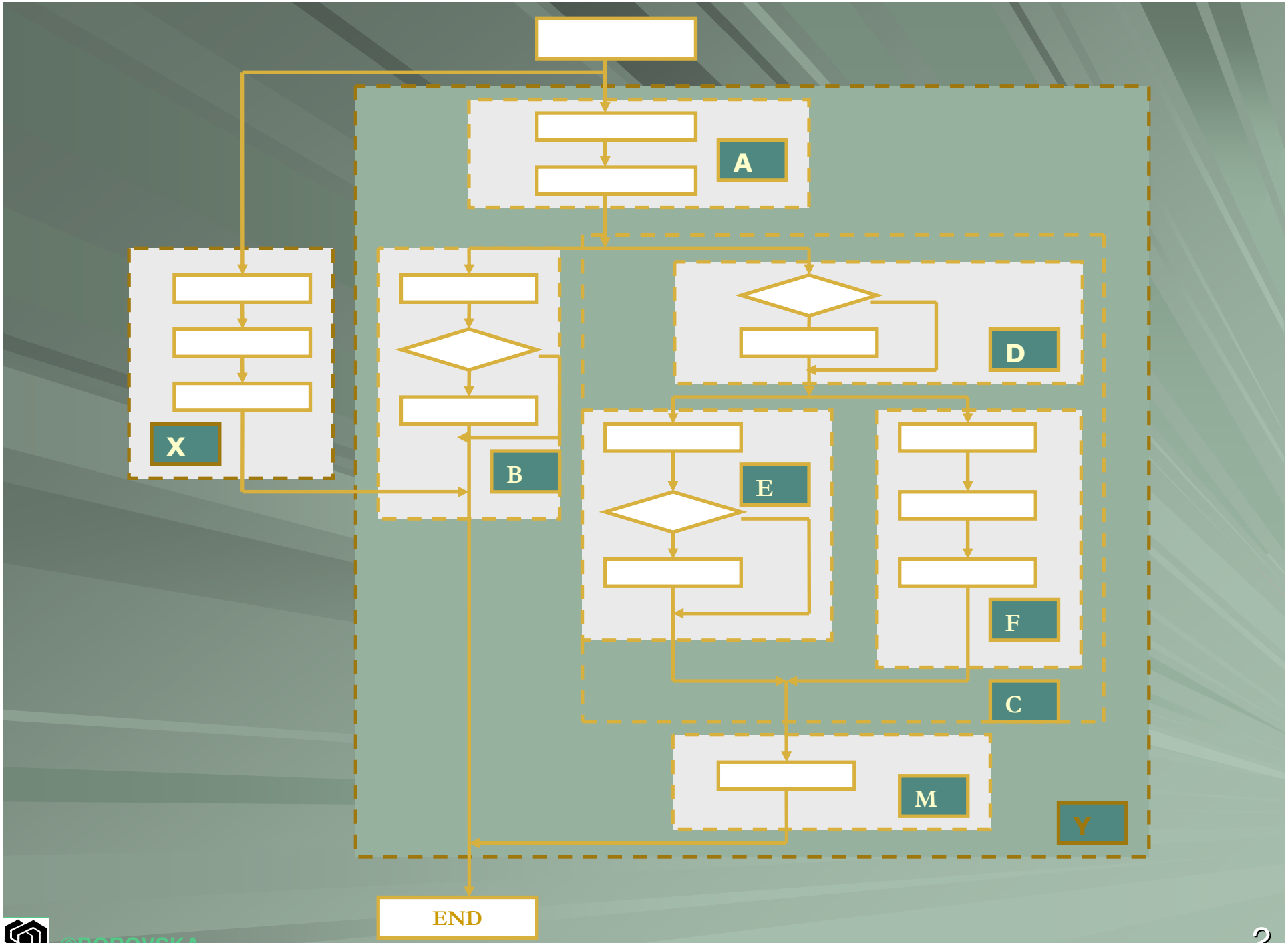
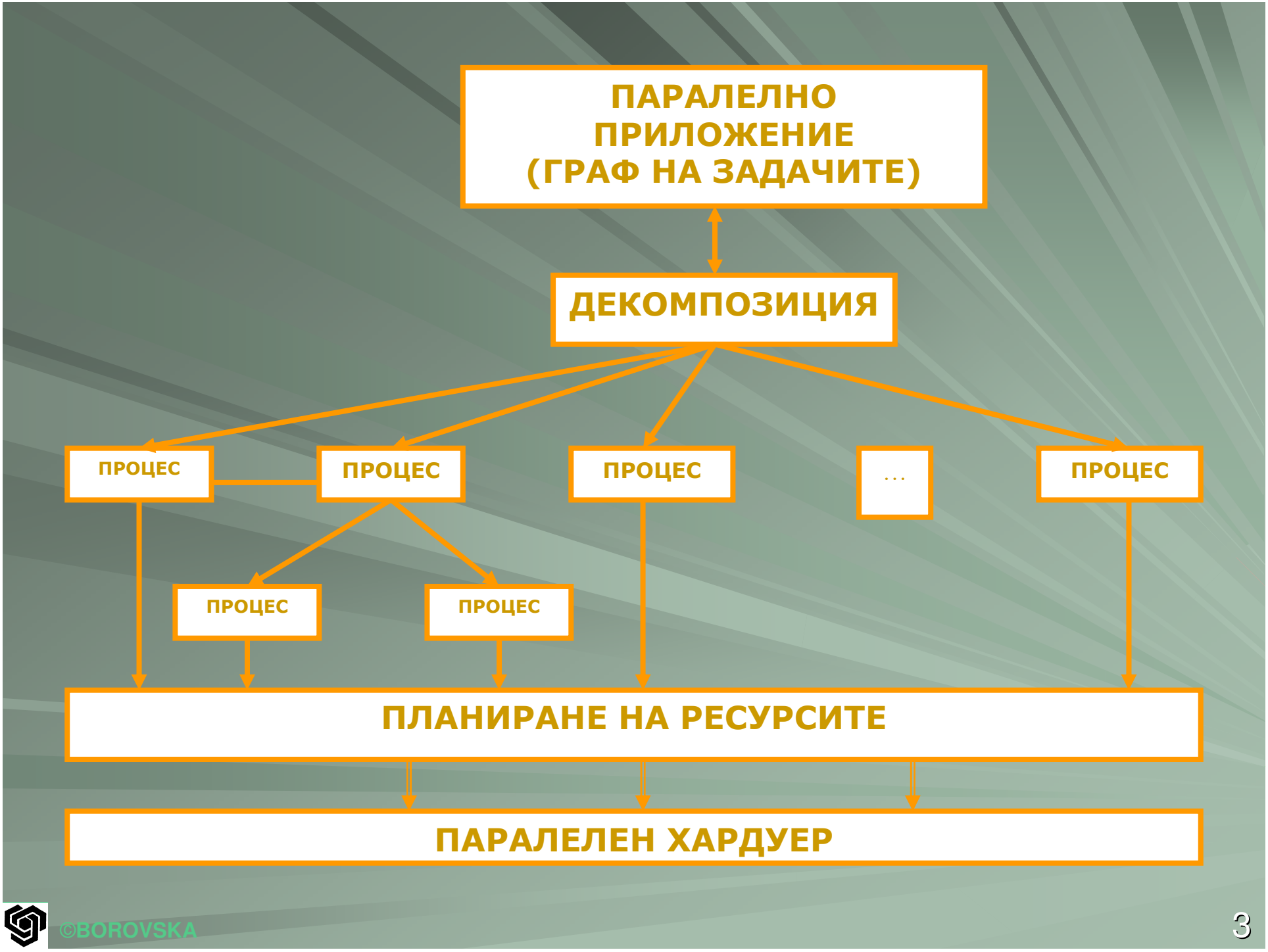
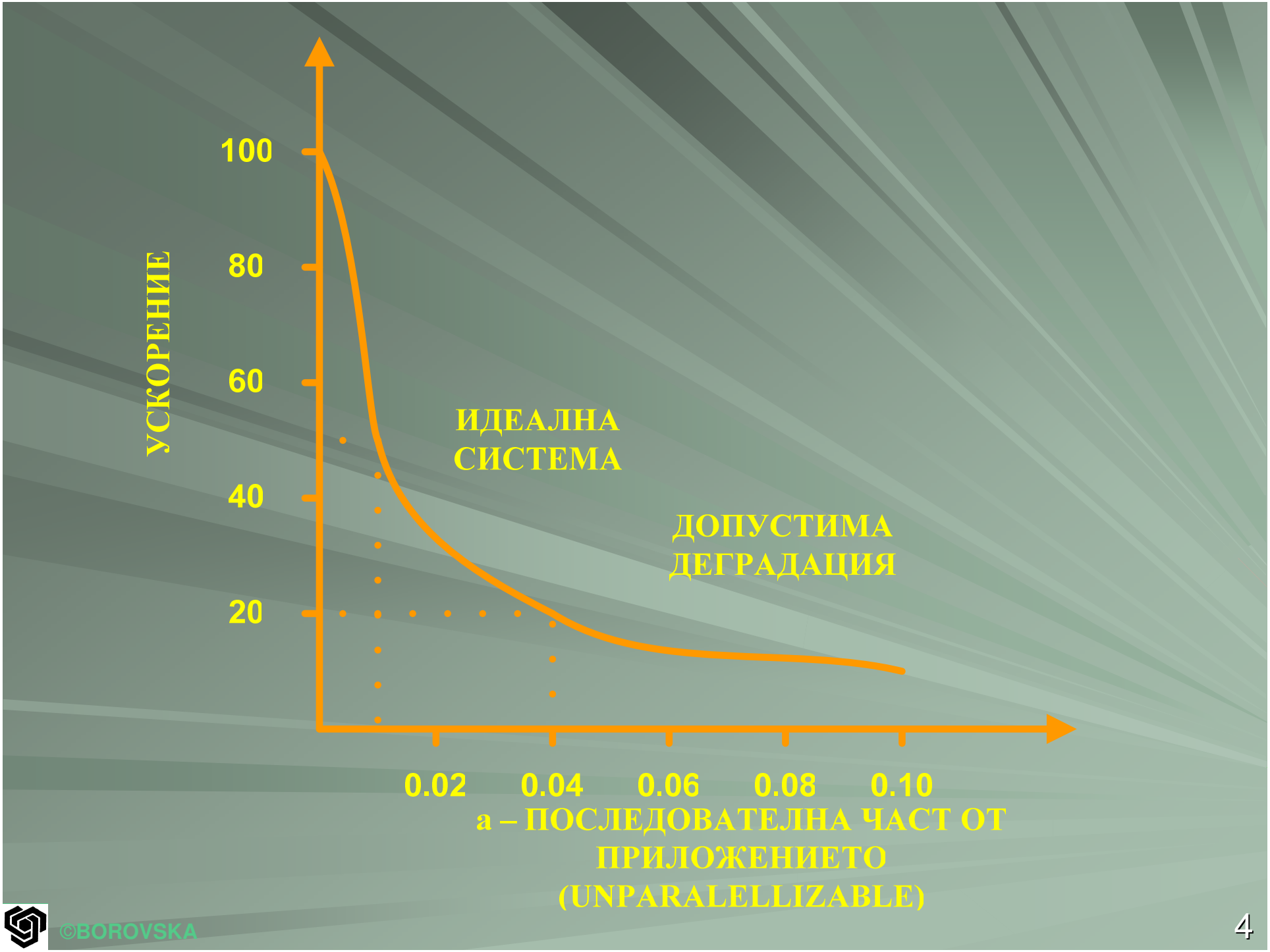
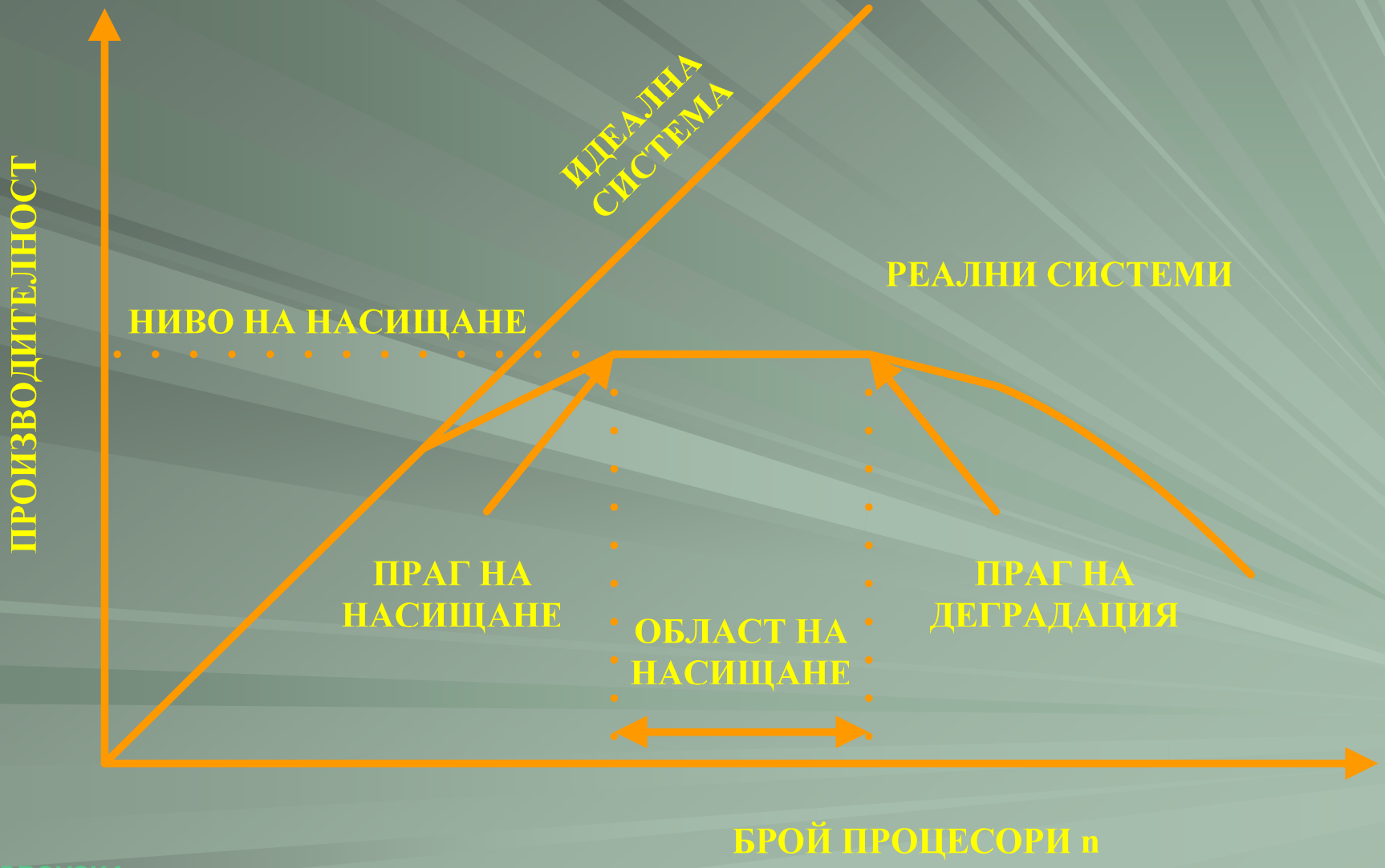
ПРОИЗВОДИТЕЛНОСТ НА ПАРАЛЕЛНИТЕ СИСТЕМИ




ПАРАЛЕЛНО ПРИЛОЖЕНИЕ (ГРАФ НА ЗАДАЧИТЕ)
ДЕКОМПОЗИЦИЯ
ПРОЦЕС
ПРОЦЕС
ПРОЦЕС
...
ПРОЦЕС
ПРОЦЕС
ПРОЦЕС
ПЛАНИРАНЕ НА РЕСУРСИТЕ
ПАРАЛЕЛЕН ХАРДУЕР


УСКОРЕНИЕ
100
80
60
40
20
ИДЕАЛНА СИСТЕМА
ДОПУСТИМА ДЕГРАДАЦИЯ
0.02
0.04
0.06
0.08
0.10
a – ПОСЛЕДОВАТЕЛНА ЧАСТ ОТ ПРИЛОЖЕНИЕТО (UNPARALELLIZABLE)

Влияние на # процесорите върху производителността



Основни мерки за производителността

- ***Мерки за работен товар и скорост***
  - *времето за изпълнение*
  - *броят на изпълнените инструкции*
  - *броят на изпълнените операции с плаваща точка*
- *Времето за изпълнение* е ***тясно обвързано със специфичната компютърна система***. То може да се промени при изпълнението на същата програма от друг компютър.

- *броят на изпълнените инструкции*, е *тясно свързан с архитектурата на системата инструкции* на процесора и не се променя, ако програмата се изпълнява на друг компютър със същата архитектура на системата инструкции
- *броят на изпълнените операции с плаваща точка, често е архитектурно независима*

Тип на работния товар	Единица за измерване на работния товар	Единица за измерване на скоростта
Време за изпълнение	Секунди (s), цикли на процесора	Приложение за секунда
Брой на инструкциите	Милиони инструкции или Милиарди инструкции	MIPS или BIPS
Брой на операциите с плаваща точка	Flop, Mflop, Gflop	Mflop/s, Gflop/s

Брой на инструкциите

- Може да се приложи за измерване на работния товар на *произволна програма*
- Работният товар се формира от броя на инструкциите, които машината е изпълнила (*динамичен брой инструкции на програмата*), а не броя на инструкциите, фигуриращи в текста на асемблерната програма (статичен брой инструкции).
- *Динамичният брой на инструкциите на програмата може да зависи от стойностите на входните данни.*

- За да се избегне зависимостта от входните данни, работният товар се оценява *за най-лошия случай на входни данни или за зададено специфично множество входни данни.*
- Дори при едни и същи входни данни, *броят на изпълнените инструкции може да варира при изпълнението на програмата на различни компютри.* Например, често RISC процесорите изпълняват от 50% до 150% повече инструкции от CISC процесорите при една и съща програма на език от високо ниво.

- *Броят на изпълнените инструкции може да варира* при изпълнението на програмата дори и на един и същ компютър, когато се използват *различни компилатори или различни оптимизационни техники*.
- По-голям брой изпълнени инструкции не означава задължително по-голямо време за изпълнение.

Време за изпълнение

- За дадена програма на определен компютър работният товар може да се дефинира като *цялото време, необходимо за изпълнение на програмата*.
- Времето за изпълнение трябва да се измерва *по стенен часовник*
- Основната измервателна единица в случая е *секунда*

Времето за изпълнение зависи от множество фактори:

- *Алгоритъмът* определя съществено времето за изпълнение
- *Структурирането на данните*
- *Входните данни*
- *Компютърната платформа:* очевидно, хардуерът и операционната система влияят върху производителността. Различно време за изпълнение може да се получи и при две компютърни платформи с еднакви процесори при една и съща тактова честота.

- Производителността не зависи само от процесора. Другите фактори са *йерархията на паметта, версията на операционната система и дали приложението използва всички компютърни ресурси или ги използва в режим на времеделене с други приложения*.
- *Езика за програмиране:* различни времена за изпълнение се получават при използване на различни езици за кодиране на приложението. Допълнително, *компилаторът и свързващият редактор също оказват значително влияние*.

Брой на операциите с плаваща точка

- ***Правила***: всяко събиране, изваждане или умножение се брои за 1 flop, обработката на индексите не се брои, присвояванията не се броят отделно, изолирано присвояване се брои като 1 flop, сравнението се брои за 1 flop, преобразуването на тип се брои за 1 flop, делението или квадратен корен се броят за 4 flop, изчисляването на синус, експонента, и др. се броят за 8 flop.

- *При използването на времето за изпълнение или броят на инструкциите като мярка за работния товар се получават следните странни ефекти: работният товар се променя при изпълнението на приложението на различни системи и още повече, работният товар може да бъде определен само чрез изпълнението на програмата.*
- Мярката с броя на операциите с плаваща точка е по-стабилна.

- ***Съвременните програми на език от високо ниво могат да постигнат от 5% до 40%-но използване на ресурсите на последователните компютри, и от 1% до 25% на паралелните компютри***
- Времето за изпълнение се измерва за конкретна компютърна платформа при строго спазване на условията за тестване включващи хардуерната платформа, опциите на компилатора, множеството на входните данни, и т.н.

Ускорение

- параметър, който показва *относителните предимства на паралелното решаване на даден проблем*
- Ускорението при n-процесорна система се определя като *отношение на времето за последователното изпълнение T_s към времето на паралелното изпълнение T_{par} на един и същ проблем*.

$$S_n = T_s / T_{par}$$

- За сравнение се избира *времето на най-добрия последователен алгоритъм* за решаването на проблема

Ефективност

- определя се като *отношение на ускорението към броя на процесорите, участващи в паралелното решаване на проблема*:

$$E_n = S_n/n$$

- *представлява мярка за полезното време, изразходвано от процесорите за изчисления*
- В реалните паралелни системи максимална ефективност не се постига ($E_n<1$) поради факта, че процесорите не използват цялото си време за полезни изчисления, а част от времето им се изразходва за поддържане на паралелизмите и взаимодействието на паралелните процеси.

Допълнителни разходи за паралелизми и взаимодействие

- времето за изпълнение на паралелната програма е:

$$T = T_{comp} + T_{par} + T_{interact}$$

Където T_{comp}, T_{par} и $T_{interact}$ представят съответно *времената на изчислителните операции, за поддържането на паралелизмите и за взаимодействията*.

- *Допълнителните разходи при паралелните програми $T_0 = T_{par} + T_{interact}$ могат да бъдат разделени на три класа:* ***дисбаланс, паралелизми, и взаимодействие (вкл. синхронизация, комуникация и агрегация).***
- Съществуват ***три типа операции за поддържане на паралелизмите, които са източник на допълнителни разходи***:
- *Управление на процесите* като създаване, унищожаване, контекстно превключване и др.
- *Операции за групиране* като създаване или унищожаване на група процеси.
  - *Операции за запитване на процесите* като запитване за идентификатора на процеса, ранга, идентификатора на групата, размера на групата и др.

операции за взаимодействие, които са източник на допълнителни разходи:

- ***Синхронизация*** като бариера, заключване, критични секции и събития.
- ***Агрегация*** като редукция и сканиране.
- ***Комуникация*** (от-точка-до-точка и колективна комуникация, четене/запис на общи променливи).
- ***допълнителните разходи за паралелизмите и взаимодействията са значителни в сравнение с базовото време за изчисления***

Често *допълнителните разходи се дължат на ядрото на операционната система или на системното програмно осигуряване*, което обуславя различни допълнителни разходи дори при идентична архитектура на процесорите при различните системи.

Така например, апаратната латентност на комутатора в SP2 е по-малък от 1 μs. При използването на IP протокол на ниво ядро, тази латентност нараства до стотици μs. Фирмата IBM е разработила специален протокол, който заобикаля ядрото и успява да намали латентността до 39 μs.

Компаниите, производители на паралелни компютри, предоставят ***данни за производителността*** в различни форми като върхова (максимална) производителност (MIPS или Mflop/s) за възел или за система с *n* възела. Те, обаче, рядко предоставят ***данни за допълнителните разходи, освен за стартиране (start-up) и пропускателна способност за комуникации от-точка-до-точка***.

Условията за измерването на допълнителните разходи включват:

- *използваните структури от данни* (желателно е използваните структури от данни да бъдат достатъчно малки за да се съдържат изцяло във възела на паметта за да се избегнат честите грешки поради липсващи страници)
- *езикът за програмиране*
- *библиотеката*
- *опцията на компилатора*
- *използвания комуникационен хардуер и протокол*
- *дали се използват показанията на стенния часовник или се отчита времето на процесора*

Методи за измерване на допълнителните разходи

- Основните проблеми са два:

1. повечето компютърни системи осигуряват отчет на сравнително големи временни интервали от порядъка на *μs* или дори *ms*
2. процесорите в паралелните компютри, *особено в масивно паралелните процесори и клъстерите*, функционират **асинхронно**, и не използват общ тактов генератор. Това предизвиква *проблема за синхронизацията по време*

- *Популярните схеми за оценка на времето за комуникация от-точка-до-точка са схемата пинг-понг и схемата на горещия картоф (нар. се още схема на “пожарната команда”)*
- Създаването на процес или група процеси изисква значителни допълнителни разходи, особено при масивно паралелните процесори и клъстерите. Така например, създаването на процес при SP2 изисква 10 000 μs или повече, което е еквивалентно на милиони flop. Създаването на група изисква около 1 ms.
- *Програма, която изисква създаването на много групи от процеси, трябва да има голям размер на гранулата.*

Закон на Амдал (фиксиран размер на проблема)

- изчислителният работен товар *W* е фиксиран
- работният товар може да се раздели на две части: $W = \alpha W + (1-\alpha)W$, където α е частта от *W*, която трябва да бъде последователно, а останалата част $1-\alpha$ може да бъде изпълнена паралелно на *n*-те възела

$$S_n \rightarrow \frac{1}{\alpha} \quad (n \rightarrow \infty)$$

$$S_n = \frac{W}{\alpha W + (1-\alpha)(W/n)} = \frac{n}{1+(n-1)\alpha}$$

Следствия от закона на Амдал:

- ***За даден работен товар максималното ускорение има горна граница от 1/α. С други думи последователната част на програмата е тясно място при паралелната обработка.***
- За да се постигне добро ускорение е важно да се направи последователното тясно място *α* по възможност най-малко.
- Когато даден проблем се състои от указаните две части (последователна и паралелна), трябва да ускорим изчисленията на по-голямата част.

да отразим в закона на Амдал допълнителните разходи при паралелната обработка

$$S_n = \frac{W}{\alpha W + (1-\alpha)(W/n) + T_0} =$$

$$= \frac{n}{1 + (n-1)\alpha + \frac{nT_0}{W}} \rightarrow \frac{1}{\alpha + \frac{T_0}{W}} \left(n \rightarrow \infty \right)$$

производителността на паралелната програма се ограничава не само от последователното тясно място, но и от средните допълнителни разходи

Закон на Густафсън (фиксирано време).

- *Джон Густафсън (1988)* предлага концепцията за фиксирано време при която се подобрява ускорението чрез мащабиране на проблема с увеличаването на размера на паралелната машина.
- ***Мащабиране за по-висока точност.***
- ***Ускорение при фиксирано време.***
- Последователното време, обаче, за обработката на мащабирания изчислителен товар е:

$$W' = \alpha W + (1 - \alpha) n W$$

Ускорението при фиксирано време за мащабиран товар се дефинира като:

$$S'_n = \frac{T'_s}{T'_{par}} = \frac{\alpha W + (1-\alpha) n W}{W} = \alpha + (1-\alpha) n$$

Ускорението при фиксирано време е линейна функция на броя на процесорите при положение, че работният товар е мащабиран по такъв начин, че да се поддържа фиксирано времето за изпълнение

Мащабирано ускорение

- Нека да отчетем допълнителните разходи при закона на Густавсън.

$$S'_n = \frac{T'_s}{T'_{par}} = \frac{\alpha W + (1-\alpha) n W}{W + T_0} = \frac{\alpha + (1-\alpha) n}{1 + T_0 / W}$$

Според обобщения закон на Густафсън линейно ускорение може да се постигне като се поддържа намаляваща функция на допълнителните разходи по отношение на n.

Закон на Сън и Ни (ограничения от паметта)

- ***Xian-He Sun и Lionel Ni (1993) са предложили модел за определяне на ускорението, който обобщава законите на Амдал и Густафсън и отчита влиянието както на броя на процесорите, така и на паметта.***
- **Практически, *много от приложенията за паралелни компютри се ограничават от капацитета на паметта*, а не от процесорите или от параметрите на В/И система. Това е особено важно за *мултикомпютри с разпределена памет*.**

Ускорение, ограничено от паметта

- *При мултикомпютрите общият капацитет на паметта нараства линейно с увеличаването на възлите в системата*. Нека с *M* да означим капацитета на паметта на един възел. За масивно паралелен процесор с *n* възела общият капацитет на паметта ще бъде *nM*.
- Паралелната част от работния товар може да се мащабира *G(n)* пъти.
- ***Факторът G(n) отразява увеличаването на работния товар при n-кратното увеличаване на капацитета на системната памет***.

- За мащабирания товар се получава

$$W = \alpha W + (1-\alpha)G(n)W$$

Ускорението, ограничавано от паметта

$$S_n^* = \frac{T_s^*}{T^*_{par}} = \frac{\alpha W + (1-\alpha)G(n)W}{\alpha W + (1-\alpha)G(n)W/n} = \frac{\alpha + (1-\alpha)G(n)}{\alpha + (1-\alpha)G(n)/n}$$

Като се отчетат допълнителните разходи за ускорението се получава:

$$S_n^* = \frac{\alpha W + (1-\alpha)G(n)W}{\alpha W + (1-\alpha)G(n)W/n + To} = \frac{\alpha + (1-\alpha)G(n)}{\alpha + (1-\alpha)G(n)/n + To/W}$$

Съществуват три специални случая за уравнението:

- *G(n)=1*. Това е случаят на фиксиран размер на проблема → еквивалентно на закона на Амдал.
- *G(n)=n*. Това е случаят, при който работният товар се увеличава n пъти при n-кратното увеличаване на паметта → еквивалентно на закона на Густафсън при фиксирано време за изпълнение.
- *G(n)>n → уравнението за ускорението, ограничено от паметта*

Модели на изопроизводителност

- Терминът *система* се използва за представяне на *двойката приложение (работен товар) и паралелна компютърна платформа.*
- Тези мерки първо се използват за да се определи функцията на мащабиране на малка система, която после се използва за да се прогнозира производителността на голяма система.

Изоефективност

- *Грама, Гупта и Кумар* предлагат мярката изоефективност за характеризиране на възможностите на системата да се мащабира.
  - Ефективността на произволна система → ***E=f(W,n)***
- *функция на изоефективността* на системата → фиксираме ефективността на някаква константа (напр. 50%) и решаваме уравнението на ефективността за работен товар *W*

- Тази функция показва съотношението между степента на нарастване на работния товар и степента на увеличаване на размера на машината за поддържането на една и съща ефективност.
- *Колкото е по-малка изоефективността, толкова са по-добри възможностите за мащабиране на системата*
- При увеличаване на размера на машината не е необходимо значително мащабиране на работния товар
- Изоефективността е полезно средство за предсказване на необходимото увеличаване на работния товар при увеличаването на размера на машината.

- Изоскорост
- Изоутилизация

$$U = P_n / (nP_{peak}) = W / (nT_nP_{peak})$$

където P_n е скоростта на паралелната система с n възела, а P_{peak} е максималната скорост.

Системи с малка изоутилизация имат по-добри възможности за мащабиране отколкото системи с голяма изоутилизация.

Аномалии

- Нормално, ускорението при паралелната обработка е по-малко от броя на процесорите т.е. $1 \leq S_n \leq n$
- На практика са известни случаи, когато ускорението е по-голямо от броя на процесорите и **такова ускорение се нарича *суперлинейно*, а явлението – *аномалия при увеличаването на ускорението (acceleration anomaly).***

- ***В случаите на суперлинейно ускорение, ефективността на паралелната система е по-голяма от 1.***
- Когато с увеличаване на броя на процесорите, ускорението се намалява (времето за решаване на проблема расте), говорим за *сублинейно ускорение*, а явлението се нарича *аномалия с намаляване на ускорението (deceleration anomaly).*

Оценка на производителността на паралелните системи на основата на фазово-паралелния модел

- Всички операции в рамките на една фаза (стъпка) трябва да завършат преди започването на следващата стъпка
- Дадена фаза C_i има изчислителен работен товар W_i *(Mflops)*, за изпълнението й от едно-процесорна система са необходими T_s секунди и се характеризира с ниво на паралелизъм L_i.
- Когато дадена фаза се изпълнява от n – процесорна система, за която $1 \le n \le L_i$, времето за паралелното й изпълнение е:

$$T_n(i) = \frac{T_s(i)}{n}$$

- ***Общото време за изпълнение T_n*** на паралелната програма от ***n*** – процесорна паралелна система се определя като:

$$T_n = \sum_{i=1}^{k} \frac{T_i(i)}{\min(L_i)} + T_0$$

- ***Скоростта на изпълнение*** на паралелната програма от *n* – процесорна паралелна система P_n се определя като отношението на изчислителния работен товар W към времето за на паралелната програма от *n* – процесорна паралелна система T_n:

$$P_n = \frac{W}{T_n}$$

- Времето за изпълнение на паралелната програма от идеална паралелна система (безкраен брой процесори, няма допълнителни разходи) се нарича ***критичен път*** и се означава с T_∞:

$$T_\infty = \sum_{i=1}^{k} \frac{T_s(i)}{L_i}$$

- Най-малката стойност на n, за която се постига $T_n = T_\infty$ се нарича ***максимален паралелизъм*** и се означава с n_{max}

$$n_{\max} = \max_{1 \le i \le k}(L_i) \qquad T_n = T_\infty$$

- n_{max} *указва максималния брой възли, които могат да бъдат използвани за намаляване на времето за изпълнение*

- ***Поддържаната скорост P_n има за горна граница максималната производителност $P_\infty = W/T_\infty$.***
- ***Времето за изпълнение на паралелната програма от n – процесорна паралелна система T_n има долна граница, определена от T_s/n и T_∞, т.е.:***

$$T_n = \max\left(T_s/n, T_\infty\right)$$

- ***Средният паралелизъм*** T_s/T_∞ осигурява горна граница за ускорението т.е.

$$S_n \geq T_s / T_\infty.$$

- ***Брент*** е доказал, че като се игнорират допълнителните разходи, *времето за изпълнение на паралелната програма от n–процесорна паралелна система T_n се ограничава от следните неравенства*:

$$(T_s/n) \le T_n < (T_s/n + T_\infty)$$

$$\max(T_s/n, T_\infty) \le T_n < (T_s/n, + T_\infty)$$

- ***Средният грануляритет*** G_{av} се дефинира като отношението $G_{av}=W/T_0$, където W е общият изчислителен товар и $T_0 = Ti_{nteract} + T_{par}$ са общите допълнителни разходи.

- Реципрочната мярка, наречена ***средни допълнителни разходи*** $T_{0,av}$ се дефинира като отношението $T_{0,av}=T_0/W$.
- ***В повечето случаи общите допълнителни разходи са доминирани от разходите за комуникация!***
  - Разпространено убеждение за паралелната обработка е, че *разходите за комуникация нарастват при използването на повече процесори*. Това, обаче, не винаги е вярно. Така например, при изпълнението на еталонните програми *APT* на паралелния компютър *SP2* разходите за комуникация намаляват при увеличаване размера на паралелната машина до 256 процесора.

Означение	Терминология	Дефиниция
T_s	Време за последователно изпълнение на програмата	$T_s = \sum_{i=1}^{k} T_s(i)$
T_n	Време за изпълнение на програмата от *n*–процесорна паралелна система	$T_n = \sum_{i=1}^{k} \frac{T_i(i)}{\min(L_i)} + T_0$
T_∞	Критичен път	$T_\infty = \sum_{i=1}^{k} \frac{T_s(i)}{L_i}$
P_n	Скорост на *n*–процесорна паралелна система	$P_n = \frac{W}{T_n}$
S_n	Ускорение на *n*–процесорна паралелна система	$S_n = \frac{T_s}{T_n}$
E_n	Ефективност на *n* –процесорна паралелна система	$E_n = \frac{T_s}{n T_n}$
U_n	Използване на ресурсите на *n*–процесорна паралелна система	$U_n = \frac{P_n}{n P_{peak}}$
T_0	Общи допълнителни разходи на *n*–процесорна паралелна система	$T_{0,av} = T_{par} + T_{interact}$
L_{av}	Среден паралелизъм	$L_{av} = \frac{T_s}{T_\infty}$
$T_{0,av}$	Средни допълнителни разходи на *n*–процесорна паралелна система	$T_{0,av} = \frac{T_0}{W}$
G_{av}	Среден грануляритет	$G_{av} = \frac{W}{T_0}$

Еталонни програми за оценка на производителността на паралелните системи

- ***Еталонните програми (benchmarks) Splash и Splash-2*** са предназначени за числени приложения и са разработени в Университета в Станфорд.
- ***Еталонната програма Perfect*** е създадена в Университета в Илиноис и се използва за оценка на различни ***паралелизиращи компилатори и техники***.

Комплектът еталонни програми (benchmark suite) ***NPB*** *(NAS Parallel Benchmark)* е разработен от програмата за симулации на числени приложения в аеродинамиката (*NAS-Numerical Aerodynamic Simulation*) *в изследователския център на NASA* за оценка на производителността на паралелни суперкомпютри.

- ***Еталонната програма NPB съдържа имплементации на код на Fortran, базиран на MPI.***
- Тя съдържа ***5 ядра*** *(EP, MG, CG, FT, IS)* и ***три програми със симулирани приложения*** *(LU, SP, BT)*.
- ***Еталонната програма EP (Embarassingly Parallel)*** оценява горната граница на производителността при обработката на числа с плаваща точка.

- Еталонната програма *CG (Conjugate Gradient method)* изчислява най-малките собствени числа на симетрична положително дефинирана матрица. Тя включва неструктурирани изчисления на решетката, изискващи нерегулярни комуникации на дълги разстояния.
- Еталонната програма *FT* решава *3D* частни диференциални уравнения с използване на спектрален метод, основан на *БПФ*, което също изисква комуникации на дълги разстояния.

- ***Еталонните програми BT (Block Tri-diagonal), LU (block lower triangular, block upper triangular) и SP (Scalar Penta-diagonal)*** прилагат различни методи за решаването на уравненията на *Navier-Stokes*.

- ***Комплектът еталонни програми PARKBENCH (PARallel Kernels and BENCHmarks) е разработен от комитета PARKBENCH, основан от група учени с интереси в областта на паралелната обработка.***
- Тези еталонни програми са предназначени за приложения за мултикомпютри с разпределена памет, кодирани с *Fortran 77* и библиотеките за обмен на съобщения *PVM* или *MPI*.

- Създадени са също така версии на еталонните програми за *Fortran 90* и *HPF*, както и за архитектури с обща памет.
- ***Комплектът PARKBENCH включва 4 класа еталонни програми:***
  - ***микроеталони на ниско ниво***, тестващи основни архитектурни параметри на паралелната система като скорост на аритметичните операции, бързодействие на паметта и на кеша, пропускателна способност на комуникационната подсистема, допълнителни разходи за синхронизация и др.

- ***Ядра***, тестващи производителността на често използвани подпрограми за числена обработка като матрична обработка, БПФ, решаване на частни диференциални уравнения, и др.
- ***Компактни еталони*** на приложения.
- ***Еталонни програми за компилатори*** на HPF. Те съдържат няколко прости синтетични приложения с конструкции за явен паралелизъм

- Комплектът еталонни програми ***Parallel STAP*** *(Parallel Space-Time Adaptive Processing)* е предназначен приложения за обработка на радарни сигнали в реално време.
- ***Най-популярният комплект еталонни програми за комерсиални приложения е TPC, разработен от Съвета за оценка на производителността при обработката на транзакции и бази данни (Transaction Processing Performance Council).***

- ***Съветът TPC осигурява спецификации на отворен стандарт за неговите еталонни програми***, които могат да бъдат използвани от всяка организация, оценяваща производителността на компютърни системи и предлага одитори, проверяващи резултатите преди публикуването им.

TPC (Transaction Processing Council) – независим от търговците

www.tpc.org/

- ***Еталонната програма TPC-C измерва производителността и отношението цена/производителност на системи за обработка на (online) транзакции. Тя е най-широко разпространената комерсиална еталонна програма за приложения и съдържа еталони за он-лайн обработка на транзакции.*** Тя симулира цялостната среда на компания, в която операторите на терминали изпълняват транзакции над база данни.

Всички TCP еталони измерват производителността в брой транзакции за секунда, като се включва изискване за времето за отговор!

- Фамилията еталонни програми *SPEC* е създадена от организацията с нестопански цели *Корпорация за стандартизация на оценките на производителността.*

SPEC (Standard Performance Evaluation Corporation) – *създава стандартизирани комплекти от еталонни приложни програми (www.spec.org)*

- ***За всяка еталонна програма и за всеки комплект от такива SPEC дефинира няколко мерки за производителността на цялата система.*** *SPEC* започва първо със създаването на еталонни програми за измерването на производителността на централния процесор, но после дейността й се разширява като обхваща изчисленията по модела клиент/сървър, комерсиални приложения, В/И системи, и т.н.

- **SPEC95** измерва производителността на централния процесор, системата памет и генерирането на кода от компилатора. Времето за работа на операционната система и В/И операции се пренебрегва.
- **SPEC периодически публикува получените резултати от измерването на производителността на различни машини.**
  - Резултатите се получават с използване на консервативни или агресивни оптимизации и се отчитат съответно като “основни” или “върхови” стойности.

- ***SPEChpc96*** измерва производителността на високо паралелни компютърни системи, изпълняващи *индустриални* приложения.
- ***SPECweb96*** измерва производителността на ***Web сървъри***, като използва работен товар, извлечен от реални *WWW* сървъри.
- ***SFS (System-level File Server)*** – измерва времето за отговор и пропускателната способност на мрежови файлови сървъри на различни нива на натоварване.
- ***SDM (System Development Multitasking)*** – измерва ефективността на системата в среда с много потребители, използващи команди на *Unix* .
- ***GPC (Graphics Performance Characterization)*** – измерва производителността при графичен режим. Еталонната програма ***PLB (Picture Level Benchmark)*** измерва “вектори за секунда” или “полигони за секунда”.

Заключение

Най-разпространеното схващане е, че времето за изпълнение е единствената надеждна мярка за измерване на компютърната производителност. *Този подход предполага изпълнението на потребителското приложение на целевата машина и измерване на изтеклото астрономическо време.* В някои случаи съществуват трудности за прилагане на този подход и освен това при него могат да бъдат допуснати *подвеждащи интерпретации*.

Основният проблем е, че само ограничен брой приложни програми са преносими на различни компютърни платформи. За изпълнението на дадена приложна програма на новосъздаден паралелен компютър са необходими *много месеци усилия за модифициране, настройка и тестване на приложния код*, да не говорим за необходимите средства за заплащане на програмистите и машинното време.

Потребителите се интересуват от следните характеристики на производи-телността: ***време за изпълнение, скорост, пропускателна способност, утилизиция, която е добър показател за ефективността на цената и отношение производителност/ цена.***

КРАЙ