

## Přednáška #1: **Analýza složitosti a škálovatelnosti paralelních algoritmů**

### Typy paralelních počítačů

**Definice 1.** *(Almasi, Gottlieb 89) Paralelní počítač je skupina **výpočetních prvků** (processing elements, (PEs)) (nebo **výpočetních uzlů** (computing nodes, CNs)), které komunikují a spolupracují, aby rychle vyřešily velké a náročné problémy (úlohy).*



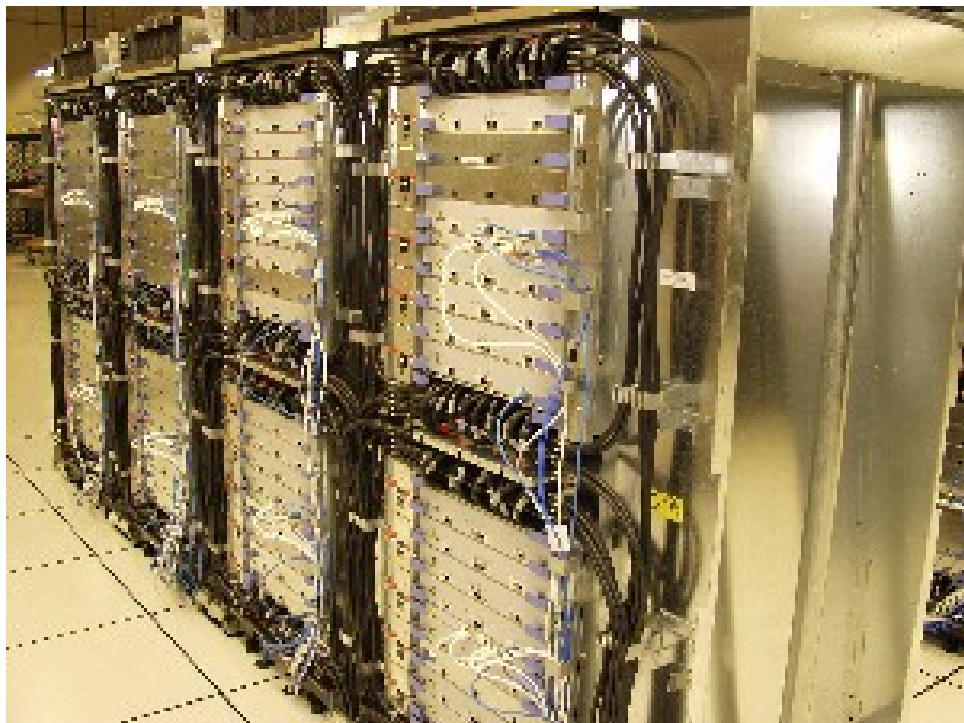
- **Multicore** processors: jednotky až desítky procesorových **jader** na 1 čipu.
- **GPU clusters**: desítky a stovky GPU (Graphical PU) připojených k CPU.
- **Desktop** multiprocessors: jednotky CPU v PC nebo pracovní stanici.
- **SMP** (Symmetric Multi-Processors) **servers**: desítky CPU se sdílenou pamětí (HP SuperDome, Sun SunFire, IBM Regatta, SGI Altix)
- **Distributed shared memory multiprocessors**: desítky CPU s virtuálně sdílenou ale fyzicky distribuovanou pamětí.
- **Clusters of workstations** (COW): svazky stovek až tisíců výpočetních uzlů (Linux svazky).
- Tightly coupled **massively parallel multiprocessors**: stovky až desetitisíce CPU se speciální propojovací sítí (IBM BlueWaters, IBM BlueGene).

- Uvažujme předpověď počasí v **oblasti** o velikosti  $3000 \times 3000 \times 11 \text{ km}^3$  na dobu **2 dnů**.
- Tato oblast je rozdělena na **segmenty** (např. metodou konečných prvků) o velikosti  $0.1 \times 0.1 \times 0.1 \text{ km}^3 \implies$  počet segmentů je řádově  $10^{11}$ .
- Parametry modelu (teplota, rychlost větru) jsou počítány s časovým krokem 30 minut.
- **Nové** hodnoty parametrů jednoho segmentu jsou počítány z **předchozích** hodnot parametrů tohoto segmentu a segmentů sousedních.
- Předpokládejme, že výpočet parametrů 1 segmentu spotřebuje 100 instrukcí.
- Pak 1 **iterace** = **aktualizace hodnot** všech parametrů v celé oblasti vyžaduje cca  $10^{11} \times 100 \times 96 \doteq 10^{15}$  operací.
- Sekvenční počítač s 1Gflop bude potřebovat  $10^6$  sekund  $\doteq$  280 hodin = 11 dní.
- To je ale pozdě, protože modelujeme počasí pro příští 2 dny.
- Paměťový problém (data se nevejdou do hlavní paměti sekv. počítače a musí být odkládána na disk) může řešení ještě mnohonásobně zhoršit!
- Pro výpočet spolehlivého modelu počasí je třeba mnoho iterací!!!!

$\implies$  rozdělení dat do pamětí mnoha PE a // zpracování s pravidelnou výměnou dat je jediné schůdné řešení.

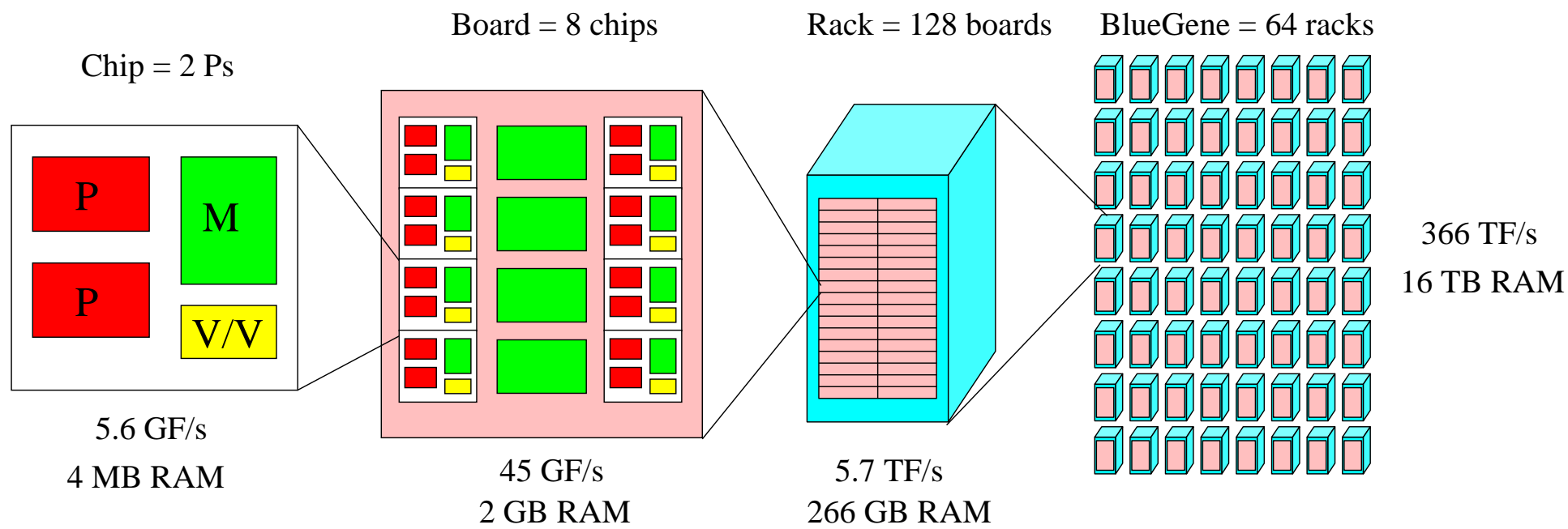
[www.top500.org](http://www.top500.org)

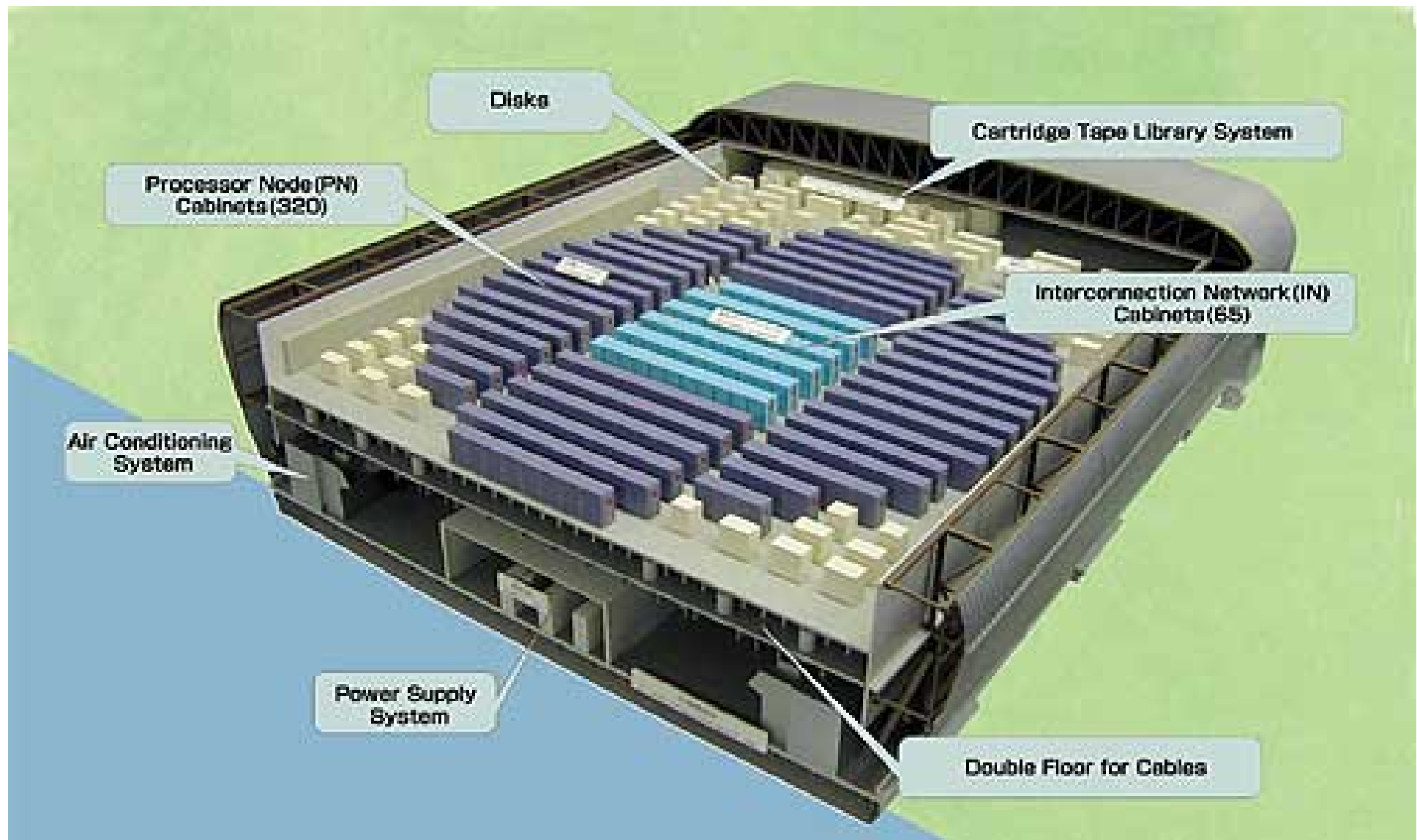
IBM Blue Gene



# IBM Blue Gene (pokr.)

4





Je-li dán problém a algoritmus pro jeho řešení, pak vzniká důležitá otázka:

Roste-li velikost problému, jak se mění chování/časová/paměťová složitost algoritmu?

⇒ nutnost znát/odvozovat asymptotické spodní a dolní meze složitostí.

## Asymptotika - definice

**Definice 2.** Necht'  $\mathcal{N}^+ =$  množina přirozených čísel a  $\mathbb{R}^+ =$  množina kladných reálných čísel. Necht'  $f, g : \mathcal{N}^+ \rightarrow \mathbb{R}^+$  jsou 2 funkce. Pak

- $f(n)$  je řádu nejvýše  $g(n)$ , psáno  $f(n) = O(g(n))$ , jestliže  

$$\overline{\exists c \in \mathbb{R}^+ \quad \exists n_0 \in \mathcal{N}^+ \quad \forall n \geq n_0 : f(n) \leq c \cdot g(n)}.$$
- $f(n)$  je řádu nejméně  $g(n)$ , psáno  $f(n) = \Omega(g(n))$ , jestliže  

$$\overline{\exists c \in \mathbb{R}^+ \quad \exists n_0 \in \mathcal{N}^+ \quad \forall n \geq n_0 : f(n) \geq c \cdot g(n)}.$$
- $f(n)$  je téhož řádu  $g(n)$ , psáno  $f(n) = \Theta(g(n))$ , jestliže  

$$f(n) = O(g(n)) \text{ a } f(n) = \Omega(g(n)).$$
- $f(n)$  je striktně nižšího řádu než  $g(n)$ , psáno  $f(n) = o(g(n))$ , jestliže  

$$\forall c \in \mathbb{R}^+ \quad \exists n_0 \in \mathcal{N}^+ \quad \forall n \geq n_0 : f(n) < c \cdot g(n).$$
- $f(n)$  je striktně vyššího řádu než  $g(n)$ , psáno  $f(n) = \omega(g(n))$ , jestliže  

$$\overline{\forall c \in \mathbb{R}^+ \quad \exists n_0 \in \mathcal{N}^+ \quad \forall n \geq n_0 : f(n) > c \cdot g(n)}.$$



**Příklad 3.** ■ Základní vztahy:  $1 = o(\log n)$ ,  $\log n = o(\sqrt{n})$ ,  $\sqrt{n} = o(n)$ .

■ Odvozené vztahy:  $n/\log n = \omega(\sqrt{n})$ ,  $n = o(n \log n)$ ,  $n \log n = o(n^2)$ .

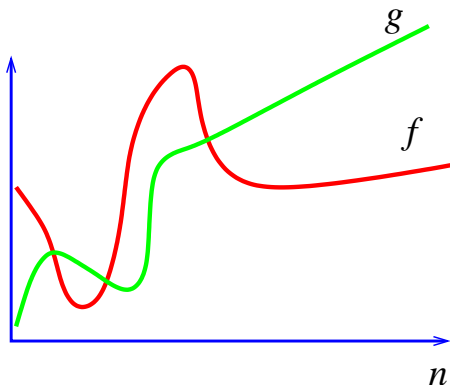
■ Jelikož  $\log n = o(n)$ , platí  $\log \log n = o(\log n)$ . Ale:  $\log n^2 = \Theta(\log n)$ .

■  $n = O(n + \log n)$  a taky  $n + 1000 \log n = O(n)$ , a tudíž  $n = \Theta(n + \log n)$ .

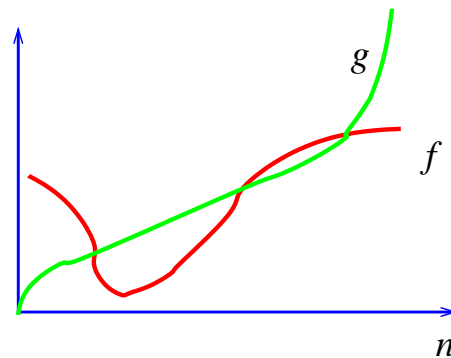
■ Stirlingova formule  $n! \doteq \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$  implikuje  $\log(n!) = \sum_{k=1}^n \log k = \Theta(n \log n)$ .

■ Jelikož  $\sum_{k=2}^n \frac{1}{k} < \int_1^n \frac{1}{x} dx < \sum_{k=1}^{n-1} \frac{1}{k}$ , dostaneme  $\ln n + \frac{1}{n} < \sum_{k=1}^n \frac{1}{k} < \ln n + 1$ . A proto,  $\sum_{k=1}^n \frac{1}{k} = \Theta(\log n)$ .

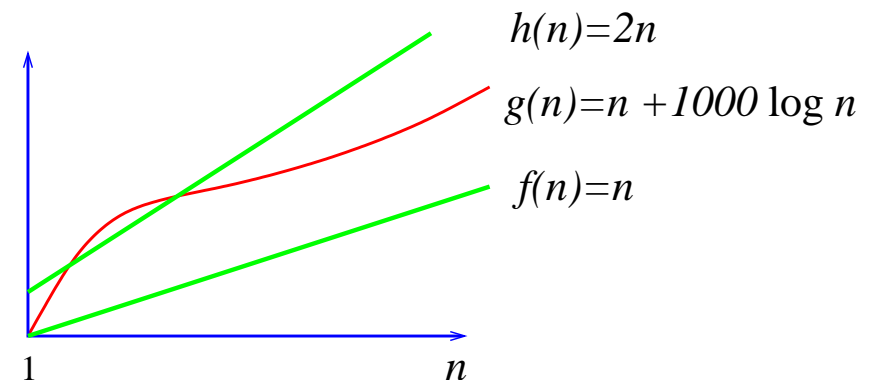
■ Je-li  $p$  konstanta, pak  $\frac{n}{2} \left(\frac{n}{2}\right)^p = \frac{n^{p+1}}{2^{p+1}} < \sum_{k=1}^n k^p < n \cdot n^p$ , a proto  $\sum_{k=1}^n k^p = \Theta(n^{p+1})$ . ♣



(a)  $f(n) = \Theta(g(n))$ .



(b)  $f(n) = o(g(n))$ .



(c)  $n + 1000 \log n = \Theta(n)$ .

<b>Transitivita:</b>	$f(n) = O(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n)).$ Podobně pro $\Omega, \Theta, o, \omega$ .
<b>Reflexivita:</b>	$f(n) = O(f(n)).$ Podobně pro $\Omega, \Theta$ .
<b>Symetrie:</b>	$f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n)).$
<b>Transpoziční symetrie:</b>	$f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n)),$ $f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n)).$
<b>Inkluze:</b>	$f(n) = o(g(n)) \Rightarrow f(n) = O(g(n)),$ $f(n) = \omega(g(n)) \Rightarrow f(n) = \Omega(g(n)).$

Zapamatujte si následující analogie:

$f(n) = O(g(n))$	$f(n) = \Omega(g(n))$	$f(n) = \Theta(g(n))$	$f(n) = o(g(n))$	$f(n) = \omega(g(n))$
$\approx$				
$a \leq b$	$a \geq b$	$a = b$	$a < b$	$a > b$

**Definice 4.**

1. Neznámá konstanta se zapisuje  $O(1)$ .
2.  $f(n)$  je polynomiální funkce, jestliže platí  $f(n) = \Theta(n^{O(1)})$ .
4.  $f(n)$  je polylogaritmická funkce, jestliže platí  $f(n) = \Theta(\log^{O(1)}(n))$ .
3.  $f(n)$  je sublineární funkce, jestliže platí  $f(n) = o(n)$ .





### Definice 5.

$T_A^K(n)$	<u>časová složitost</u> / <u>doba výpočtu</u> sekv. algoritmu $A$ , který řeší problém $K$ na vstupních datech velikosti $n$ (měří se <u>čítáním výpočetních kroků</u> / <u>instrukcí</u> ).
$SL^K(n)$	<u>spodní mez</u> časové složitosti jakéhokoli sekv. algoritmu pro řešení problému $K$ =nejhorší časová složitost <u>nejlepšího možného</u> sekv. algoritmu pro řešení $K$ . <u>Triviální spodní mez</u> je dána velikostí množiny <u>vstupních</u> ( <u>výstupních</u> ) dat $n$ .
$SU^K(n)$	<u>horní mez</u> časové složitosti pro řešení problému $K$ (=nejhorší časová složitost <u>nejrychlejšího známého</u> sekv. algoritmu pro $K$ .)

### Optimální sekvenční algoritmy

**Definice 6.** ■  $A$  je (asymptoticky) optimální sekv. alg. pro řešení problému  $K$ , jestliže platí

$$T_A^K(n) = \Theta(SU^K(n)) = \Theta(SL^K(n)).$$

■  $A$  je nejlepší známý sekv. alg. pro řešení  $K$ , jestliže platí

$$T_A^K(n) = \Theta(SU^K(n)) = \omega(SL^K(n)).$$



**Příklad 7.** Nechť  $K_1$  = problém třídění posloupnosti  $n$  čísel pomocí binární operace porovnání. Pak

$$SL^{K_1}(n) = \Omega(n \log n) \quad (\text{minimální hloubka binárního stromu s } n! = \Theta(n^n) \text{ listy})$$

$$SU^{K_1}(n) = O(n \log n) \quad (\text{MergeSort, HeapSort, QuickSort})$$

**Příklad 8.** Nechť  $K_2$  = problém násobení matic  $A_{n,n} \times B_{n,n}$ . Pak

$$SL^{K_2}(n) = \Omega(n^2) \quad (\text{triviální spodní mez})$$

$$SU^{K_2}(n) = O(n^q), \quad 2 < q < 3 \quad (\text{Strassen } q = 2.81, \text{ Coppersmith-Winograd } q = 2.376)$$

V porovnání se **sekvenčními** algoritmy zde  $\exists$  navíc 1 parametr/dimenze:

počet procesorů  $p$ .

Přirozeným cílem při paralelním řešení úloh je dosáhnout

lineárního zrychlení:

Jestliže stoupne počet procesorů  $k$  krát, chceme, aby výpočetní čas klesnul  $k$  krát.

- Tento cíl je obecně velmi obtížně splnitelný.
- Pro dosažení potřebné rychlosti/efektivnosti/zrnitosti je nutno mezi  $p$  a  $n$  udržovat určitou závislost, např.
  - $p = n^x$ , kde  $0 < x < 1$ ,
  - $p = \frac{n}{\log n}$ ,
  - $p = \log^2 n$ .

**Definice 9.**  $T(n, p)$  je čas, který uplynul od začátku paralelního výpočtu do okamžiku, kdy poslední (nejpomalejší) procesor skončil výpočet. ♣

### Diskuze

- $T(n, p)$  závisí na architektuře paralelního počítače



hodnocení výkonnosti paralelního algoritmu musí vždy brát v úvahu architekturu počítače.

- $T(n, p)$  je měřen čítáním:

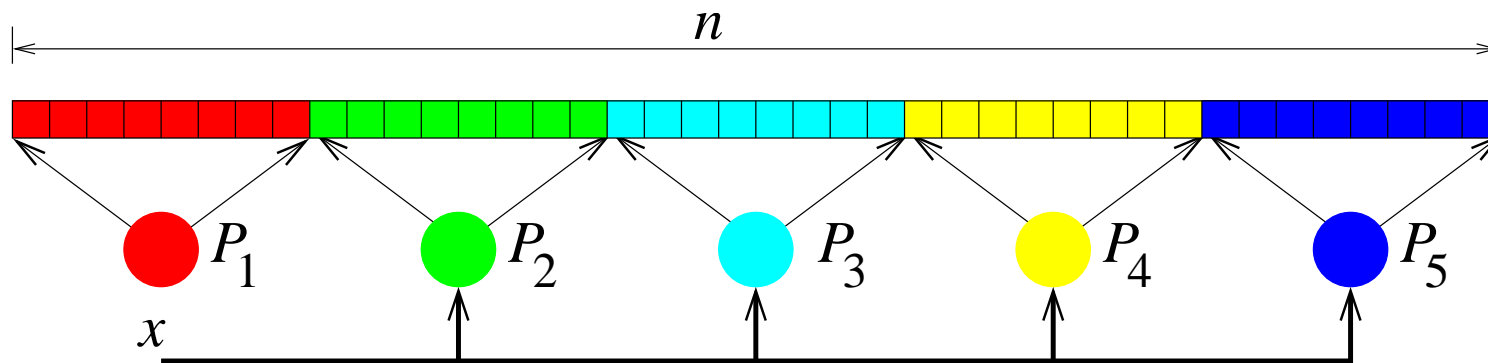
1. výpočetních kroků: aritmetické, logické, paměťové operace

+

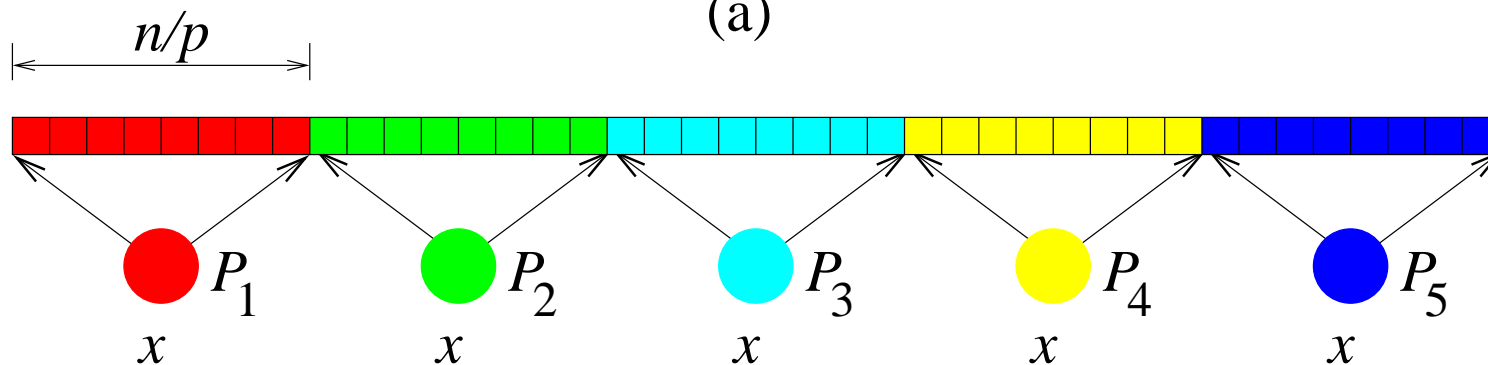
2. komunikačních kroků: přenos a výměna dat mezi procesory.

**Příklad 10.** Paralelní vyhledávání položky  $x$  pomocí  $p$  procesorů v neseřazeném vstupním souboru  $n$  různých položek uložených v sdílené paměti. Předpokládáme, že v daném okamžiku smí k dané buňce sdílené paměti přistupovat nejvýše 1 procesor.

$$T(n, p) = \underbrace{O(\log p)}_{\text{(a) rozeslání } x} + \underbrace{O(n/p)}_{\text{(b) porovnávání}} + \underbrace{O(1)}_{\text{(c) zapsání výsledku}}$$



(a)



(b)

**Definice 11.**

$$S(n, p) = \frac{SU(n)}{T(n, p)} \leq p.$$

**Lineární zrychlení****Definice 12.**

$$S(n, p) = \Theta(p).$$

**Superlineární zrychlení**

- Způsobeno určitými charakteristikami HW, které staví sekvenční algoritmy do nevýhody.
  - Typická situace: algoritmus je paměťově náročnější, než je kapacita paměti na 1-procesorovém systému, kdežto souhrnná kapacita pamětí paralelního systému stačí.
- Vzniká díky anomáliím při paralelním prohledávání kombinatorického stavového prostoru (podrobněji příští přednáška).

**Definice 13.**

$$L(n, p) = \frac{SL(n)}{p}.$$



**Příklad 14.** Spodní mez na čas paralelního třídění  $n$  čísel s  $p = n$  procesory je

$$L(n, n) = \frac{\Omega(n \log n)}{n} = \Omega(\log n).$$



**Definice 15.**

$$C(n, p) = p \times T(n, p).$$



$C(n, p)$  se také nazývá součin procesory x čas.

**Lemma 16.**

$$C(n, p) = \Omega(SU(n)).$$

**Cenově optimální algoritmus****Definice 17.**

$$C(n, p) = O(SU(n)).$$



Z Lemmatu 15 pak plyne:

$$C(n, p) = \Theta(SU(n)).$$



**Definice 18.** *(Práce synchronního systému) Necht'  $\tau = T(n, p)$  a  $p_i = \#$  procesorů aktivních (pracujících) v kroku  $i \in \{1, \dots, \tau\}$  paralelního výpočtu. Pak*

$$W(n, p) = p_1 + p_2 + \dots + p_\tau.$$

**Definice 19.** *(Práce asynchronního systému) Necht'  $T_i \leq T(n, p) = \#$  kroků provedených procesorem  $i \in \{1, \dots, p\}$  během paralelního výpočtu. Pak*

$$W(n, p) = T_1 + T_2 + \dots + T_p.$$

**Lemma 20.**

$$SU(n) \leq W(n, p) \leq C(n, p).$$

**Pracovně optimální algoritmus**

**Definice 21.**

$$W(n, p) = O(SU(n)).$$



- $C(n, p)$  zahrnuje nečinnost procesorů, kterou  $W(n, p)$  nezapočítává.
- V praxi je užitečný spíše parametr  $C(n, p)$ , protože nečinné procesory ve většině systémů není možné před dokončením celého výpočtu uvolnit pro jiný výpočet.

**Definice 22.**

$$E(n, p) = \frac{SU(n)}{C(n, p)}.$$



**Lemma 23.**  $E(n, p) = \underline{\text{zrychlení na procesor:}}$

$$E(n, p) = \frac{SU(n)}{C(n, p)} = \frac{S(n, p) \times T(n, p)}{p \times T(n, p)} = \frac{S(n, p)}{p}.$$



**Lemma 24.** *Algoritmus je (asymptoticky)*

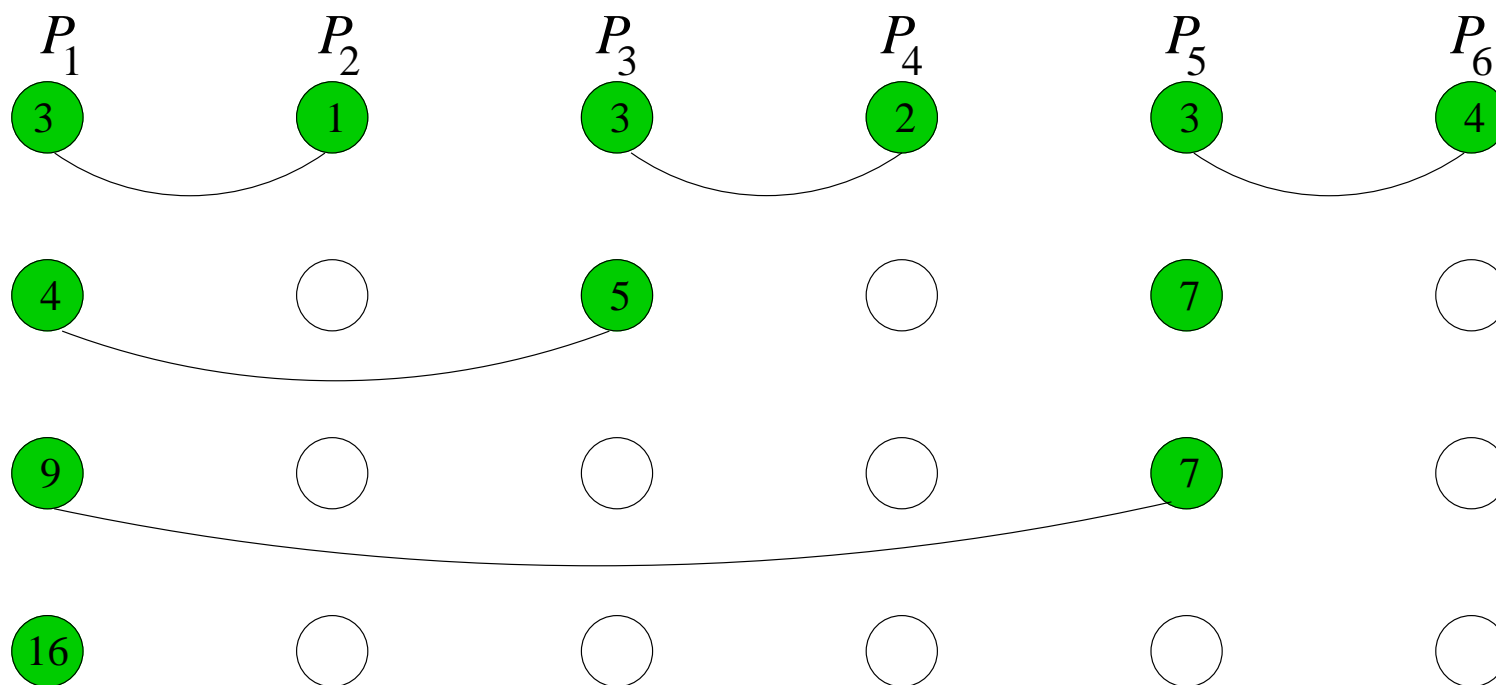
cenově optimální  
 $\Longleftrightarrow$   
*má* lineární zrychlení  
 $\Longleftrightarrow$   
*má* konstantní efektivnost.



**Příklad 25.** (Paralelní  $\Sigma$ ). Vypočtete  $\sum_{i=1}^n a_i$  čísel  $a_1, \dots, a_n$  na paralelním počítači s úplně propojenými  $n$  procesory  $P_1, \dots, P_n$ . Předpokládejte:

- jednotkový-časový model:  
čili sečíst 2 čísla na 1 procesoru a poslat číslo z 1 procesoru na druhý trvá čas 1;
- na počátku má procesor  $P_i$  číslo  $a_i$  ve svém registru.

Algoritmus PARADD:



- $SU(n) = SL(n) = \Theta(n)$
- $T(n, n) = \Theta(\log n)$
- $C(n, n) = \Theta(n \log n)$
- $W(n, n) = \Theta(n)$
- $S(n, n) = \Theta(n / \log n)$
- $E(n, n) = \Theta(1 / \log n)$

Diskuze výsledků:

- PARADD  $n$  čísel na  $n$  procesorech je pracovně optimální, ale není cenově optimální.
- Intuitivní vysvětlení: velmi malé využití procesorů. Počet procesorů, které vykonávají užitečnou práci, klesá exponenciálně rychle.

### **Zdroje neefektivnosti obecně**

1. Nedostatek užitečné práce (příliš procesorů na málo práce).
2. Velké komunikační zpoždění v porovnání s výpočetní rychlostí  
⇒ některá data by bylo lepší počítat lokálně, než o ně žádat a čekat, až dorazí.
3. Příliš velká režie synchronizace (slabá koordinace).
4. Špatná distribuce práce (nerovnoměrné rozdělení práce).

## Řešení?

### 1. Technologické:

- rychlejší komunikační HW,
- zmenšení SW komunikační režie,
- překrývání výpočetních a komunikačních kroků.

### 2. Algoritmické:

- optimální mapování algoritmu na paralelní architekturu (statické),
- vyvažování zátěže (dynamické),
- respektování škálovatelnosti problému, tzn.:

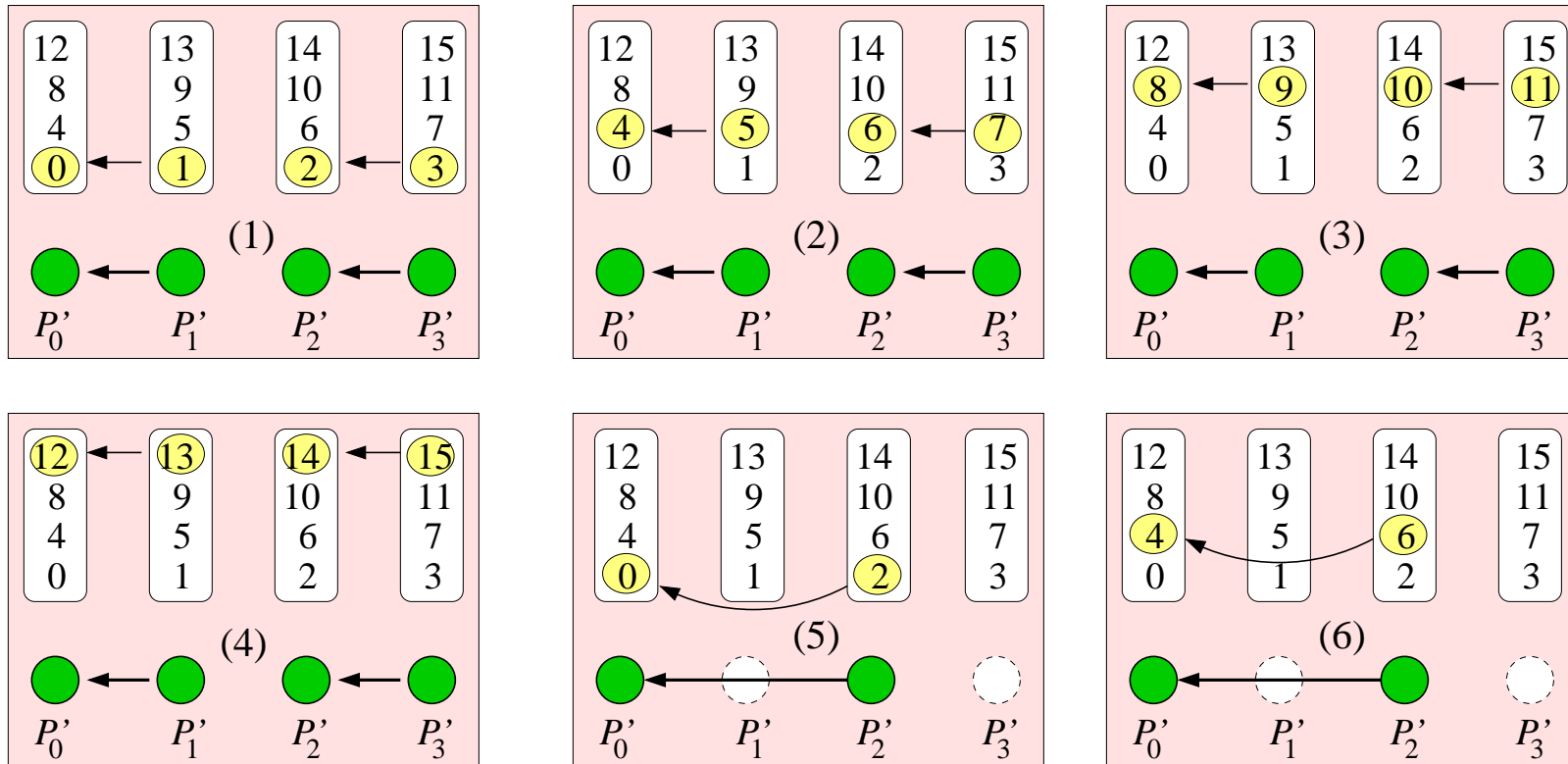
Škálovat  $p$  s  $n$  tak, aby  $\forall$  procesory byly vytíženy užitečnými výpočty většinu času



Řešení problému škálovatelnosti = hledání vhodné závislosti

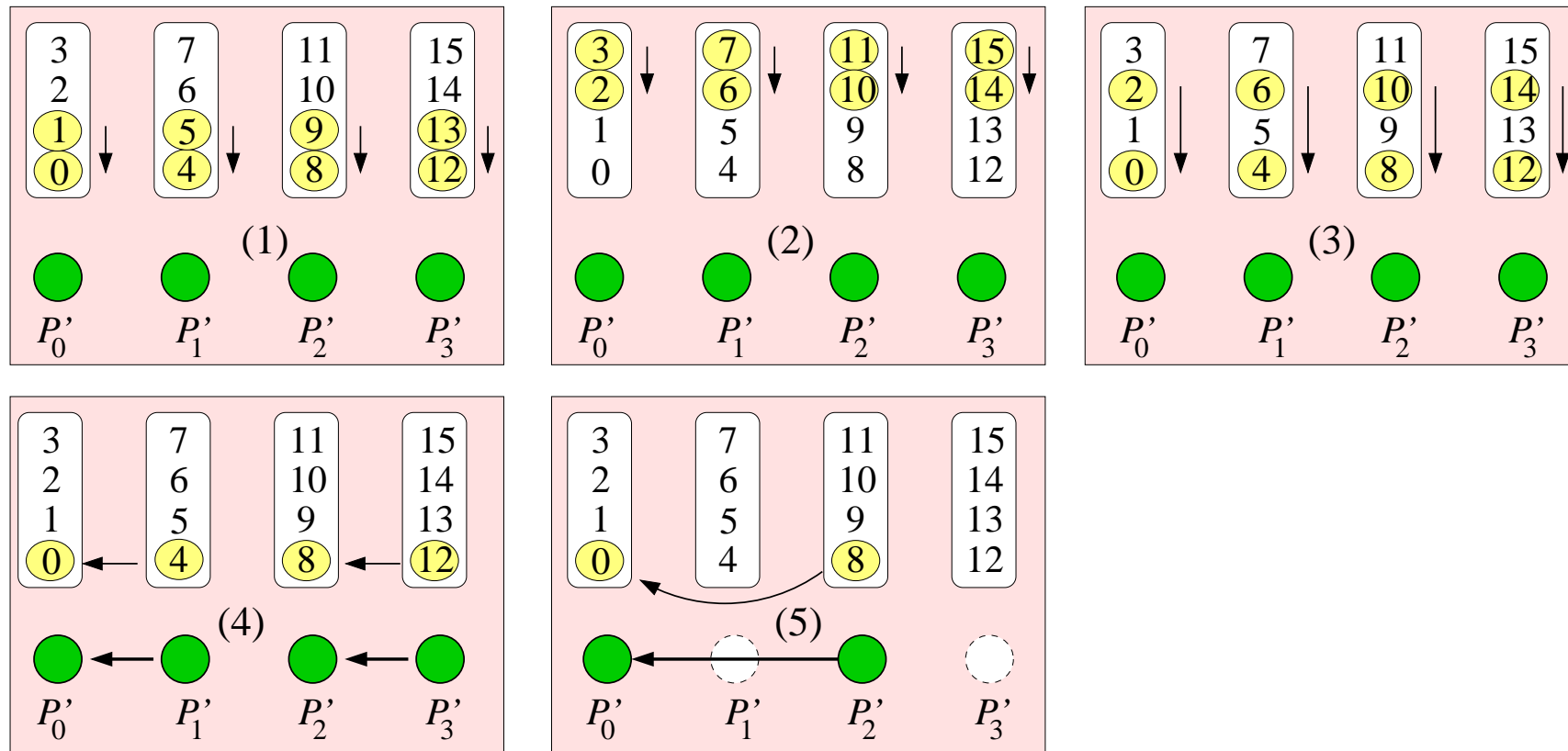
$$p \stackrel{?}{=} f(n)$$

Simulace algoritmu na méně procesorech, než je stupeň paralelismu algoritmu.



ROWSIMPARADD: Simulace  $n$ -procesorové paralelní redukce na  $p$  procesorech,  $p|n$ .

- $\log p$  fází o  $4n/p$  krocích.
- Na konci zbývá 1 procesor, který provede poslední fázi v  $4\frac{n}{p}$  krocích.
- Proto  $T(n, p) \doteq 4\frac{n}{p} \log p + 4\frac{n}{p} = \Theta(\frac{n}{p} \log p)$ .
- $C(n, p) = \Theta(n \log p), \quad S(n, p) = \Theta(p / \log p), \quad E(n, p) = \Theta(1 / \log p), \quad W(n, p) \doteq 5n.$



COLSIMPARADD: Alternativní simulace  $n$ -procesorové paralelní redukce na  $p$  procesorech,  $p|n$ , pomocí Brentova plánování.

- $T(n, p) = 4\left(\frac{n}{p} - 1\right) + 2 \log p = \Theta\left(\frac{n}{p} + \log p\right)$ .
- $C(n, p) = \Theta(n + p \log p)$ ,  $S(n, p) = \Theta\left(\frac{np}{n + p \log p}\right)$ ,  $E(n, p) = \Theta\left(\frac{n}{n + p \log p}\right)$ .

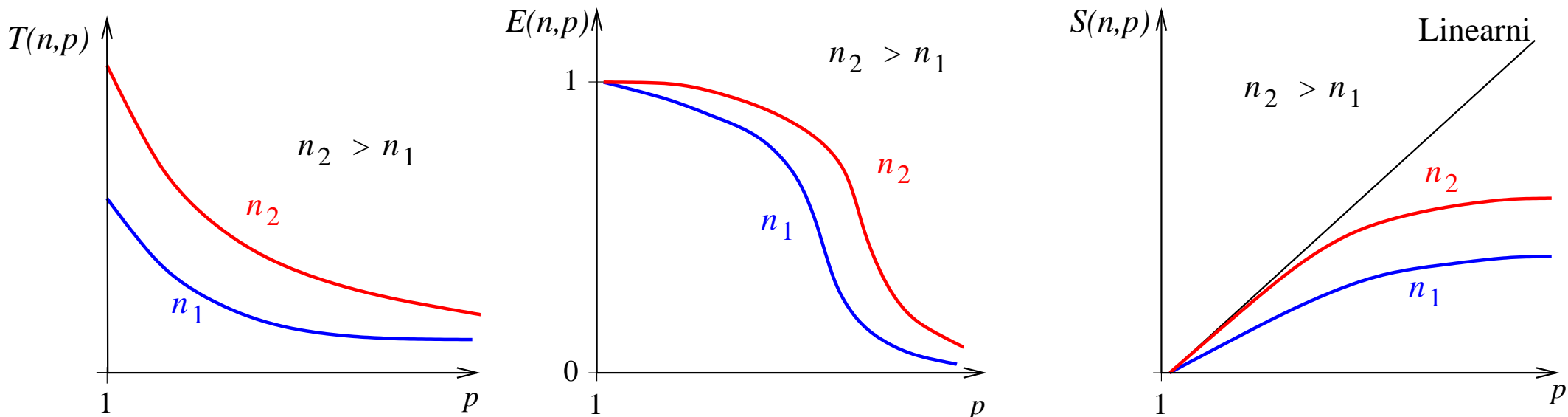
Škálovatelnost = schopnost efektně využít rostoucí počet procesorů.

= schopnost udržet efektivnost či dobry čas při

$\uparrow n$ , jestliže  $\uparrow p$   
 $\downarrow n$ , jestliže  $\downarrow p$   
 $\uparrow p$ , jestliže  $\uparrow n$   
 $\downarrow p$ , jestliže  $\downarrow n$

## Pozorování

- Jestliže  $p = 1$ , efektnost je nejlepší, ale čas je nejhorší.
- Jestliže  $p$  roste, čas klesá, ale jen do určité meze  
 $\implies$  začne klesat efektnost.





**Definice 26.** Je-li dána konstanta  $0 < E_0 < 1$ , pak izoefektivní funkce

- $\psi_1(p)$  je asymptoticky minimální funkce taková, že  $\forall n_p = \Omega(\psi_1(p)) : E(n_p, p) \geq E_0$ .
- $\psi_2(n)$  je asymptoticky maximální funkce taková, že  $\forall p_n = O(\psi_2(n)) : E(n, p_n) \geq E_0$ .

### Diskuze

- Funkce  $\psi_1(p)$  a  $\psi_2(n)$  jsou vzájemně inverzní.
- Číselně vyjadřují, jak  $n$  musí růst s  $p$ , aby se efektivnost neměnila.
- Odrážejí schopnost paralelního algoritmu udržet konstantní efektivnost (a tudíž lineární zrychlení).
- Pomalu rostoucí  $\psi_1(p)$  svědčí o dobré škálovatelnosti (pro účelné využití nově přidaných procesorů stačí zvětšit velikost problému o malý přírůstek).
- Strmě rostoucí  $\psi_1(p)$  svědčí o špatné škálovatelnosti.

**Příklad 27.** (Paralelní  $\Sigma$ ). Vypočtete  $\sum_{i=1}^n a_i$  čísel  $a_1, \dots, a_n$  na plně propojeném paralelním počítači s  $p < n$  procesory  $P_1, \dots, P_p$ . Předpokládáme model jednotkového času.

**Řešení:**

1. Každý procesor má přiděleno  $\lceil n/p \rceil$  vstupních čísel.
2. Každý procesor čte čísla z paměti a přičítá je k registru:  $2 \lceil n/p \rceil - 1$  kroků.
3.  $p$  procesorů sečte pomocí paralelní binární redukce  $p$  částečných součtů v  $\lceil \log p \rceil$  iteracích, 1 iterace = 2 kroky.

Tudíž,

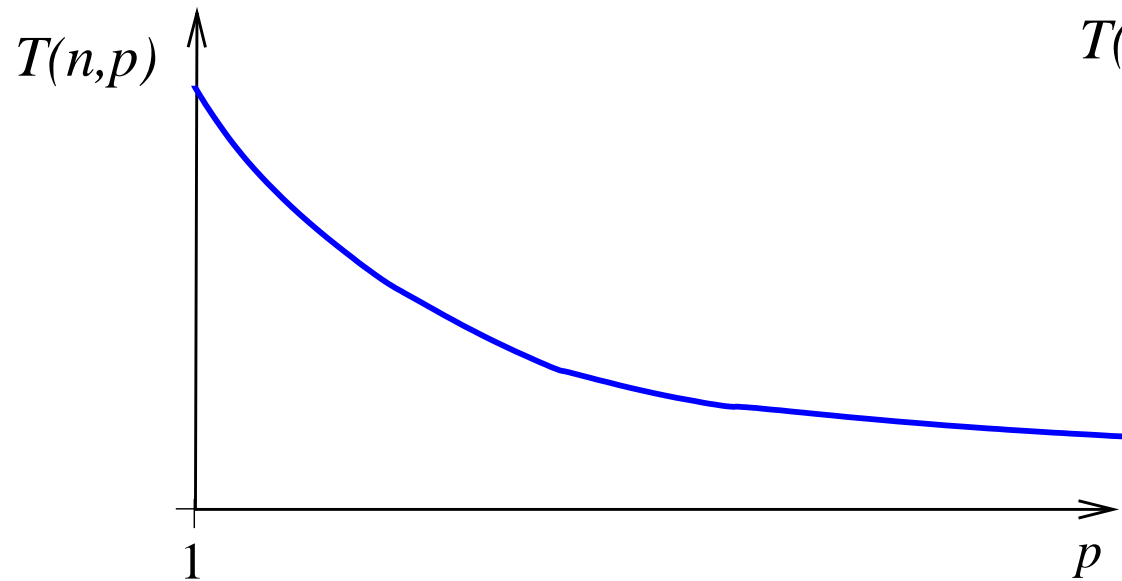
$$T(n, p) = 2 \lceil n/p \rceil - 1 + 2 \lceil \log p \rceil \doteq 2n/p + 2 \log p$$

a protože  $SU(n) = 2n$ , je

$$C(n, p) = 2n + 2p \log p \quad \text{a} \quad S(n, p) = \frac{np}{n + p \log p} \quad \text{a} \quad E(n, p) = \frac{n}{n + p \log p}.$$

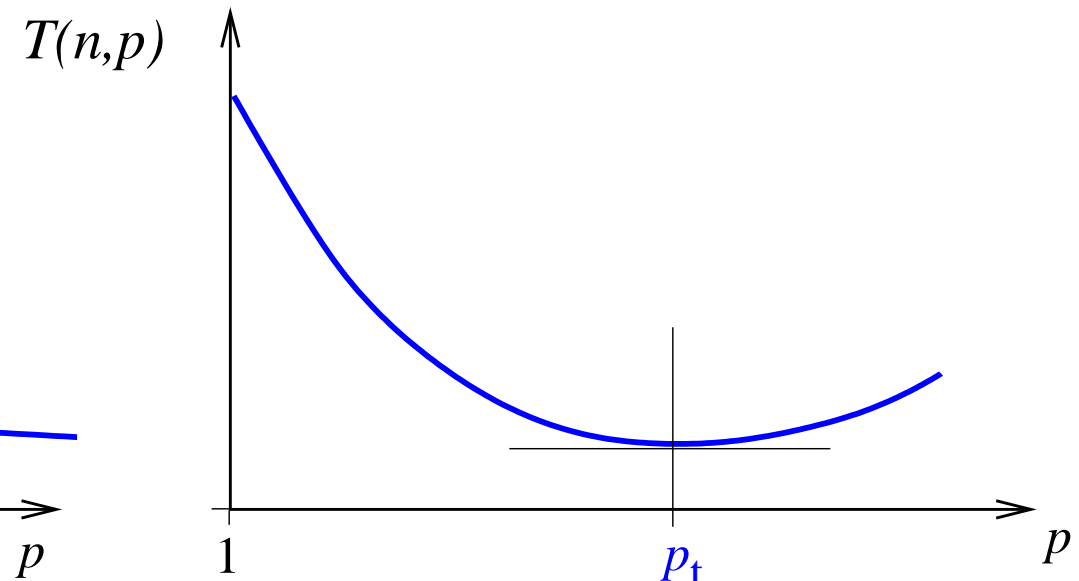
$$\psi_1(p) = p \log p \quad \text{a} \quad \psi_2(n) = \frac{n}{\log \beta n}, \quad \text{kde } \beta = \frac{1 - E_0}{E_0}.$$

**Definice 28.** Je-li dáno  $n$ , pak  $p_t$  je nejmenší počet procesorů, pro které  $T(n, p_t) =$  absolutně minimální čas  $T_{\min}$ .



(a)

Varianta (a)  $\implies p_t = p_{\max}$



(b)

Varianta (b)  $\implies \left. \frac{dT(n,p)}{dp} \right|_{p=p_t} = 0$

**Příklad 25.** (pokračování) V případě algoritmu PARADD se jedná o případ (b).

Pro

$$T(n, p) = \frac{2n}{p} + 2 \log p = \frac{2n}{p} + \frac{2 \ln p}{\ln 2}$$

má rovnice

$$\frac{dT(n, p)}{dp} = -\frac{2n}{p^2} + \frac{2}{p \ln 2} = 0$$

řešení:

$$p_t = n \ln 2 \doteq 0.6n,$$

čili

$$n/2 < p_t < n.$$

Proto

$$T_{\min} \doteq 2 \log n + 1.8.$$

Protože

$$T(n, n/2) = T(n, n) = \lceil T_{\min} \rceil,$$

volíme

$$p_t = n/2.$$

**Příklad 26.** Paralelní redukce na plně propojeném počítači, kde

- aritmetické a paměťové operace trvají čas 1,
- kdežto přenos 1 čísla mezi 2 sousedními procesory trvá čas  $k \gg 1$ .

Pak

$$T(n, p) = 2n/p + (k + 1) \log p,$$

a proto

$$p_t = \frac{2 \ln 2}{k + 1} n.$$

Například: pro  $k = 100$  je  $p_t = n/73$ .

**Kompromisy mezi rychlostí a efektivností**

- Nejrychlejší algoritmus je obecně neefektivní.
- Otázka 1: Dokáže daný algoritmus řešit úlohu současně rychle a efektivně?
- Otázka 2: Jaký počet procesorů stačí pro dosažení řádově optimálního času?

$\psi_3(n)$  je asymptoticky minimální funkce taková, že

$$\forall p = \Omega(\psi_3(n)) \quad \& \quad p = O(p_t) : T(n, p) = O(T_{\min})$$

**Příklad 25.** (pokračování) Protože

$$T(n, p) = \frac{2n}{p} + 2 \log p,$$

dostáváme

$$\psi_3(n) = \frac{n}{\log n}.$$

Nyní ale vidíme, že

$$\psi_2(n) \doteq \psi_3(n).$$

To ale znamená, že

pro  $p = \Theta(n / \log n)$  je **asymptoticky optimální čas i efektivnost.**

- Nestárnoucí postřeh z počátků éry paralelních počítačů.
- Speciální aplikace obecné zákonitosti.
- Každý výpočet se skládá z přirozeně sekvenční části (kterou může provádět pouze 1 procesor) a z přirozeně paralelní části.
- Předpokládejme, že sekvenční výpočet trvá normovaný čas  $1 = f_s + f_p$ , kde  $f_s$  = přirozeně sekvenční a  $f_p$  = přirozeně paralelní část.
- Pak  $T(n, p) \geq f_s + \frac{f_p}{p}$ , a proto

$$S(n, p) \leq \frac{1}{f_s + \frac{1-f_s}{p}}.$$

- Je zřejmé, že  $\lim_{p \rightarrow \infty} S(n, p) \leq \frac{1}{f_s}$ .
- Např., jestliže  $f_s = 10\%$ , pak  $S(n, p) \leq 10$ .
- Obecně: Pokud se nezvětšuje velikost úlohy, přidávání dalších procesorů nemá smysl, protože se saturuje paralelismus úlohy.

- Když se masivně paralelní počítače staly široce dostupné, ukázalo se, že existují paralelní algoritmy, mající přirozeně sekvenční část konstantní (V/V operace, inicializace), kdežto přirozeně paralelní část může lineárně škálovat s počtem procesorů.
- Pak doba výpočtu paralelní části zůstává nezměněná, a stejně tak celkový paralelní čas.
- Můžeme bez ztráty obecnosti předpokládat jednotkový paralelní čas:  $T(n, p) = f_s + f_p = 1$ .
- Jestliže takový algoritmus provedeme sekvenčně, bude trvat  $T(n, 1) = f_s + pf_p$  (jeden procesor simuluje práci  $p$  procesorů).
- Pak je zrychlení

$$S(n, p) = f_s + pf_p = f_s + p(1 - f_s) = p(1 - f_s + f_s/p).$$

- Tudíž  $\lim_{p \rightarrow \infty} S(n, p) = p(1 - f_s)$ .

### Závěr

Masivně paralelní počítače mohou být optimálně využity pro lineárně škálovatelné problémy.