O Desafio do Projeto de Algoritmos Paralelos

Siang Wun Song
Universidade Federal do ABC (UFABC) e Universidade de São Paulo (USP)

ERAD-SP 2011

São José dos Campos - 27 a 29 de julho de 2011

Computação Paralela - Oportunidade

Com a computação paralela, pretende-se resolver um problema com o uso de vários processadores para reduzir o tempo de execução.

Hoje a Computação Paralela já é regra e não exceção.

- Sistemas de computação maciçamente paralelos são cada vez mais comuns.
- Clusters Beowulf baseados em arquiteturas abertas tornam o uso da Computação Paralela cada vez mais popular.
- Uso cada vez mais difundido de processadores tipo multi-core, many-core e GPUs (e.g. GF100 Fermi com 512 cores).

O computador mais veloz do mundo - em junho 2011

- K Computer - Fujitsu - (Kei significa 10^{16} = 10 quadrilhões)
- total de 548.352 processadores ou cores
- 68.544 Sparc64 VIIIfx CPU, cada uma com 8 cores
- LINPACK: 8,162 PFLOPS - Velocidade de pico: 8,773 PFLOPS

Curiosidade

- K denota um ideograma chinês = 10^{16} para indicar 10 PetaFLOPS.

Costumamos ver prefixos como

- Mega = $2^{20} \cong 10^{6}$
- Giga = $2^{30} \cong 10^{9}$
- Tera = $2^{40} \cong 10^{12}$
- Peta = $2^{50} \cong 10^{15}$
- ...
- Yotta = $2^{80} \cong 10^{24}$

Curiosidade:

Será que após esse ideograma chinês que denota 10^{16}, há outros ideogramas para representar números ainda maiores?

Curiosidade

- K denota um ideograma chinês = 10^{16} para indicar 10 PetaFLOPS.

Costumamos ver prefixos como

- Mega = $2^{20} \cong 10^{6}$
- Giga = $2^{30} \cong 10^{9}$
- Tera = $2^{40} \cong 10^{12}$
- Peta = $2^{50} \cong 10^{15}$
- ...
- Yotta = $2^{80} \cong 10^{24}$

Curiosidade:
Será que após esse ideograma chinês que denota 10^{16}, há outros ideogramas para representar números ainda maiores?

Grandes números na numeração chinesa

京 = 10^16
垓 = 10^20
秭 = 10^24
穰 = 10^28
溝 = 10^32
澗 = 10^36
正 = 10^40
載 = 10^44
極恒河沙 = 10^48
阿僧只 = 10^52
那由他 = 10^56
不可思議 = 10^60
無量 = 10^64
大數 = 10^68

Grandes números na numeração chinesa

Mais curiosidades: numeração chinesa antiga, de origem budista:

Ideograma	n	10^(7x2^n)
俱胝	0	10^7
阿庾多	1	10^14
那由他	2	10^28
頻波羅	3	10^56
矜羯羅	4	10^112
阿伽羅	5	10^224
最勝	6	10^448
摩婆羅	7	10^896
阿婆羅	8	10^1792
.....		
.....		
不可量	117	10^116307449631180138879083117774530150 4
不可量轉	118	10^232614899262360277758166235549060300 8
不可說	119	10^465229798524720555516332471098120601 6
不可說轉	120	10^930459597049441111032664942196241203 2
不可說不可說	121	10^186091919409888222206532988439248240 64
不可說不可說轉	122	10^372183838819776444413065976878496481 28
不可說不可說不可說	123	10^744367677639552888826131953756992962 56

Fonte: http://zh.wikipedia.org/%E4%B8%AD%E6%96%87%E6%95%B0%E5%AD%97

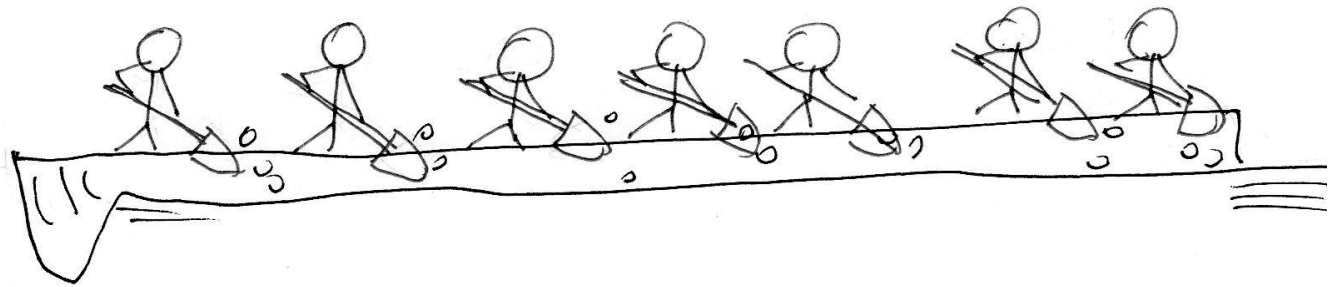
Computação Paralela - Desafios

- Como projetar um algoritmo paralelo eficiente para resolver um problema usando milhares (ou mais) de processadores?
- Alguns problemas são facilmente paralelizáveis. Outros podem demandar um esforço considerável para paralelizar, ou mesmo impossíveis de obter ganhos com o paralelismo.

  Vejamos alguns exemplos.

Algumas tarefas são facilmente paralelizáveis

Cavar uma canaleta. Fácil: basta dividir em trechos e cada um cuida de um trecho. Todos trabalham em paralelo.

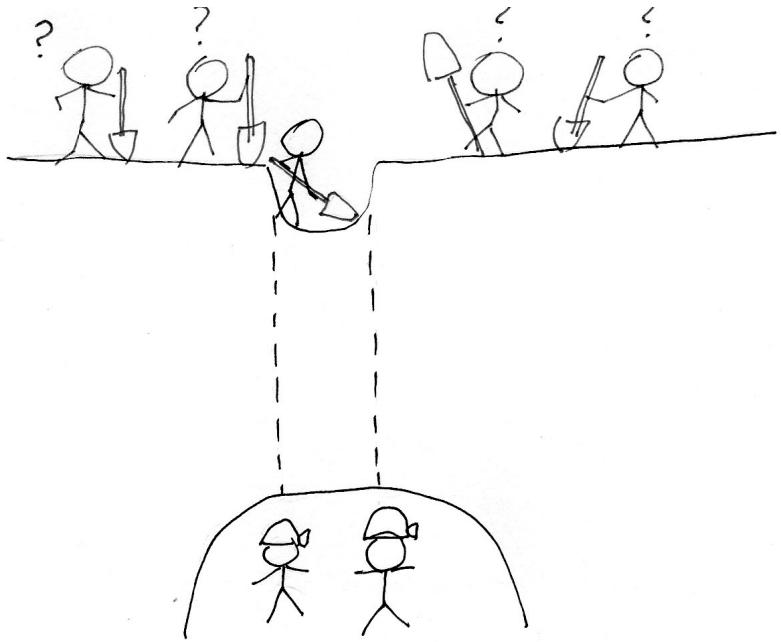


Algumas tarefas são facilmente paralelizáveis

Estação CPTM Tamanduateí - 22 de março de 2011:

Nem sempre é fácil paralelizar uma tarefa

Cavar um túnel ou um poço (e.g. para salvar mineiros presos em uma mina). A tarefa é inerentemente sequencial: uma parte de terra só pode ser retirada após retirar a parte de cima.



Importante conhecer um algoritmo eficiente sequencial

- Uma maneira é dividir o problema em subproblemas menores do mesmo tipo e cada processador resolve um subproblema executando um algoritmo sequencial.
- Às vezes isso não basta para resolver o problema e os resultados parciais obtidos em cada processador precisam ser combinados para obter a solução do problema.
- Outras vezes resolver subproblemas menores do mesmo tipo em cada processador não é o melhor modo de aproveitar os processadores e um novo modo precisa ser usado.

Vejamos alguns exemplos.

Cada processador resolve um subproblema menor pode funcionar bem

- Exemplo 1: Uma parede retangular grande precisa ser pinta com tinta branca. Têm-se centenas de pintores de parede.
  - Cada pintor pega um pedaço retangular pequeno e pinto-o de branco.
    Funcionou! Mas há exemplos que não funciona. Vejamos a seguir.

Cada processador resolve um subproblema menor pode funcionar bem

- Exemplo 1: Uma parede retangular grande precisa ser pinta com tinta branca. Têm-se centenas de pintores de parede.
  - Cada pintor pega um pedaço retangular pequeno e pinto-o de branco.
    Funcionou! Mas há exemplos que não funciona. Vejamos a seguir.

Cada processador resolve um subproblema menor nem sempre funciona

- Exemplo 2: Deseja-se fazer uma cadeira gigante de 20 metros de altura. Empregam-se centenas de carpinteiros.

Cada processador resolve um subproblema menor nem sempre funciona

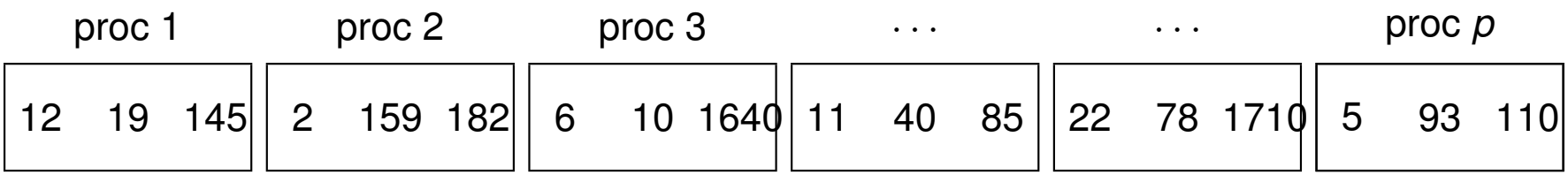
- Cada carpinteiro fabrica uma cadeira de tamanho normal não resolve: Impossível combinar as cadeiras normais em uma gigante.

Cada processador resolve um subproblema menor nem sempre funciona

- Exemplo 3: Deseja-se ordenar um conjunto de n números. Usam-se p processadores.
  - Cada processador recebe $1/p$ dos números e coloca-os em ordem. Só isso não resolve o problema. Combinar as p ordenações obtidas em cada processador pode demandar muita comunicação.

Cada processador resolve um subproblema menor nem sempre funciona

- Exemplo 3: Deseja-se ordenar um conjunto de n números. Usam-se p processadores.
  - Cada processador recebe $1/p$ dos números e coloca-os em ordem. Só isso não resolve o problema. Combinar as p ordenações obtidas em cada processador pode demandar muita comunicação.



Onde está a dificuldade?

O maior desafio para projetar um algoritmo paralelo eficiente é como evitar ou reduzir a necessidade de comunicação entre os processadores.

- Quando voce faz um trabalho de casa individual, voce o faz sem ficar falando sozinho :-)
- Um algoritmo sequencial, isto é, executado num só processador, não precisa se comunicar com os outros processadores.
- Entretanto, se vários colegas devem fazer um trabalho em grupo, voces precisam conversar entre si, para ver quem faz o que, acompanhar o que os outros já fizeram, etc.
- Num algoritmo paralelo, com poucas exceções, os processadores trocam informações entre si, dependendendo naturalmente do problema que estão resolvendo.

Onde está a dificuldade?

O maior desafio para projetar um algoritmo paralelo eficiente é como evitar ou reduzir a necessidade de comunicação entre os processadores.

- Quando voce faz um trabalho de casa individual, voce o faz sem ficar falando sozinho :-)
- Um algoritmo sequencial, isto é, executado num só processador, não precisa se comunicar com os outros processadores.
- Entretanto, se vários colegas devem fazer um trabalho em grupo, voces precisam conversar entre si, para ver quem faz o que, acompanhar o que os outros já fizeram, etc.
- Num algoritmo paralelo, com poucas exceções, os processadores trocam informações entre si, dependendendo naturalmente do problema que estão resolvendo.

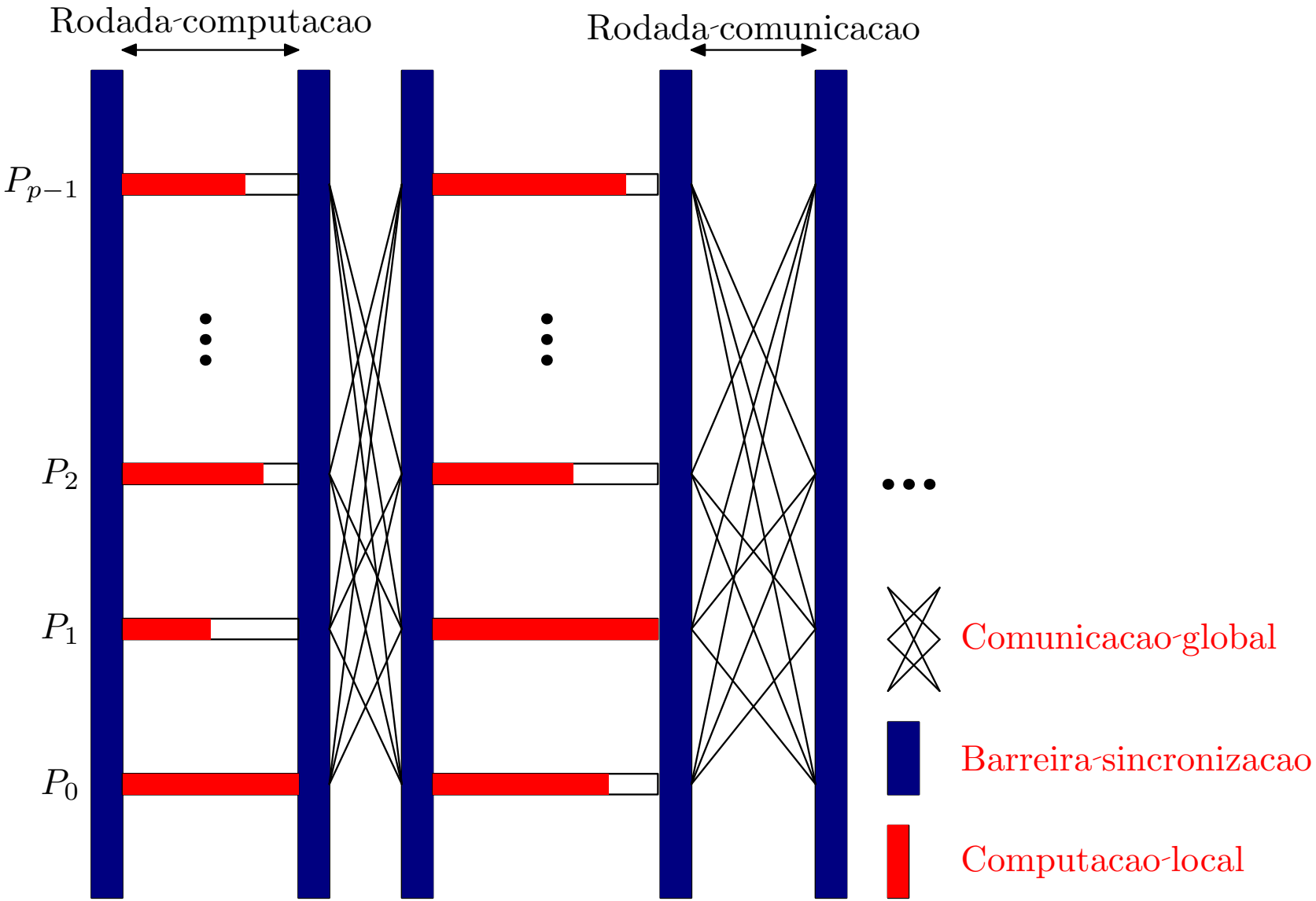
Onde está a dificuldade?

O maior desafio para projetar um algoritmo paralelo eficiente é como evitar ou reduzir a necessidade de comunicação entre os processadores.

- Quando voce faz um trabalho de casa individual, voce o faz sem ficar falando sozinho :-)
- Um algoritmo sequencial, isto é, executado num só processador, não precisa se comunicar com os outros processadores.
- Entretanto, se vários colegas devem fazer um trabalho em grupo, voces precisam conversar entre si, para ver quem faz o que, acompanhar o que os outros já fizeram, etc.
- Num algoritmo paralelo, com poucas exceções, os processadores trocam informações entre si, dependendendo naturalmente do problema que estão resolvendo.

Modelo CGM - Coarse Grained Multicomputer

- Considere a entrada do problema de tamanho $O(n)$.
- Multicomputador com p processadores, cada um com memória local de tamanho $O(n/p)$.
- Um algoritmo CGM consiste numa alternância de duas rodadas:
  - Rodada de computação: cada processador computa independentemente dos demais.
  - Rodada de comunicação: cada processador pode enviar/receber $O(n/p)$ dados dos demais.
- A meta é minimizar o número de rodadas requeridas, a fim de obter um *speed-up* linear em p.


Rodada-computacao
Rodada-comunicacao
P_{p-1}
P_2
P_1
P_0
Comunicacao-global
Barreira-sincronizacao
Computacao-local

Exemplo de projeto de um algoritmo paralelo

Problema: Dada uma sequência de números (positivos e negativos), obter a subsequência contígua de maior soma.

Mais formalmente:

Dada uma sequência de números $S = (x_1, x_2, \ldots, x_n)$, com pelo menos um número positivo,

obter uma subsequência contígua $M = (x_i, \ldots, x_j)$ com máxima soma $\Sigma_{k=i}^{j} x_k$.

Obs: se todos os números são positivos, então $M = S$.

- O problema tem aplicação em Biologia, na análise de sequências de DNAs e proteínas. A cada amino-ácido é atribuído um número positivo ou negativo.

Exemplo de projeto de um algoritmo paralelo

Problema: Dada uma sequência de números (positivos e negativos), obter a subsequência contígua de maior soma.

Mais formalmente:

Dada uma sequência de números $S = (x_1, x_2, \ldots, x_n)$, com pelo menos um número positivo,

obter uma subsequência contígua $M = (x_i, \ldots, x_j)$ com máxima soma $\Sigma_{k=i}^{j} x_k$.

Obs: se todos os números são positivos, então $M = S$.

- O problema tem aplicação em Biologia, na análise de sequências de DNAs e proteínas. A cada amino-ácido é atribuído um número positivo ou negativo.

Exemplo de projeto de um algoritmo paralelo

Problema: Dada uma sequência de números (positivos e negativos), obter a subsequência contígua de maior soma.

Mais formalmente:

Dada uma sequência de números $S = (x_1, x_2, \ldots, x_n)$, com pelo menos um número positivo,

obter uma subsequência contígua $M = (x_i, \ldots, x_j)$ com máxima soma $\Sigma_{k=i}^{j} x_k$.

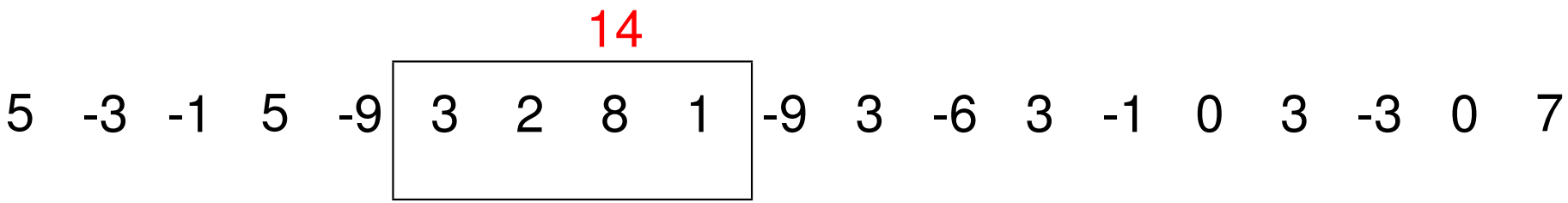
Obs: se todos os números são positivos, então $M = S$.

- O problema tem aplicação em Biologia, na análise de sequências de DNAs e proteínas. A cada amino-ácido é atribuído um número positivo ou negativo.

Exemplo: Dada a sequência

5 -3 -1 5 -9 3 2 8 1 -9 3 -6 3 -1 0 3 -3 0 7

Obter a subsequência contígua de soma máxima



Um algoritmo sequencial eficiente

5 -3 -1 5 -9 3 2 8 1 -9 3 -6 3 -1 0 3 -3 0 7

- Há um total de $n^2/2$ ou $O(n^2)$ subsequências contíguas possíveis numa sequência de n números.
- Se testarmos todas as subsequências para escolher aquela de maior soma, então teremos um algoritmo de tempo $O(n^2)$.
- Bates & Constable [1985] e Bentley [1986] apresentam um algoritmo eficiente que leva tempo $O(n)$. Vamos apresentar este algoritmo eficiente e elegante.

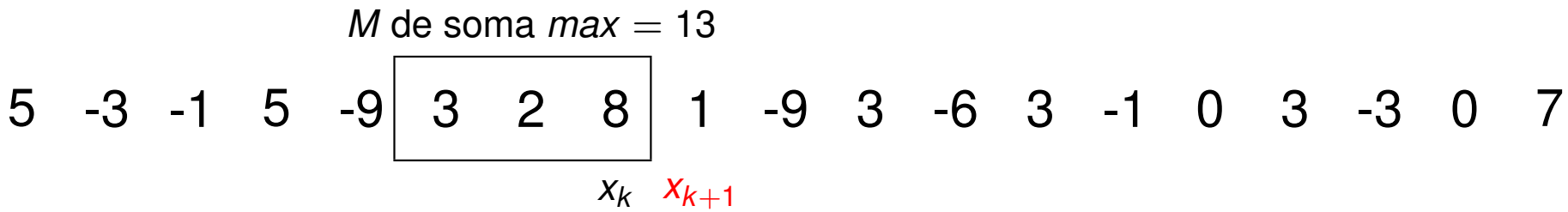
Um algoritmo sequencial eficiente

- O algoritmo sequencial eficiente varre a sequência de n números da esquerda para a direita, um número de cada vez, desde x_1 até x_n.
- Suponha que de x_1 até x_k, a subsequência máxima M obtida tenha soma max.
- Vamos considerar o próximo número x_{k+1}.

Se x_k pertence a subsequência máxima M, então é fácil decidir: se $x_{k+1} > 0$, então x_{k+1} deve ser adicionado a M, senão deixamos M inalterado.

Um algoritmo sequencial eficiente

- O algoritmo sequencial eficiente varre a sequência de n números da esquerda para a direita, um número de cada vez, desde x_1 até x_n.
- Suponha que de x_1 até x_k, a subsequência máxima M obtida tenha soma max.
- Vamos considerar o próximo número x_{k+1}.



Se x_k pertence a subsequência máxima M, então é fácil decidir: se $x_{k+1} > 0$, então x_{k+1} deve ser adicionado a M, senão deixamos M inalterado.

Sufixo máximo

- Se x_k não pertence a subsequência M de soma max, então precisamos de um conceito novo: sufixo máximo.

Seja dada a subsequência $x_i, \ldots, x_j$.

$$\underbrace{x_i \; x_{i+1} \cdots \; \cdots}_{} \; \overbrace{\boxed{x_s \; x_{s+1} \cdots \; \cdots \; x_j}}^{S}$$

S é sufixo máximo se possui o maior valor de $suf = x_s + x_{s+1} + \cdots + x_j$.

Se $suf < 0$ então fazemos S = vazio.

Exemplo de sufixo máximo

Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$

S com $suf = 5$

5	-3	-1	5	-9	3	2	8	1	-9	3	-6	3	-1	0	3	-3	0	7
x_k																		

Exemplo de sufixo máximo

Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$

S com $suf = 2$

5	-3	-1	5	-9	3	2	8	1	-9	3	-6	3	-1	0	3	-3	0	7
	x_k																	

Exemplo de sufixo máximo

Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$

S com $suf = 1$

5	-3	-1	5	-9	3	2	8	1	-9	3	-6	3	-1	0	3	-3	0	7
		x_k																

Exemplo de sufixo máximo

Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$

S com $suf = 6$

5	-3	-1	5	-9	3	2	8	1	-9	3	-6	3	-1	0	3	-3	0	7
			x_k															

Exemplo de sufixo máximo

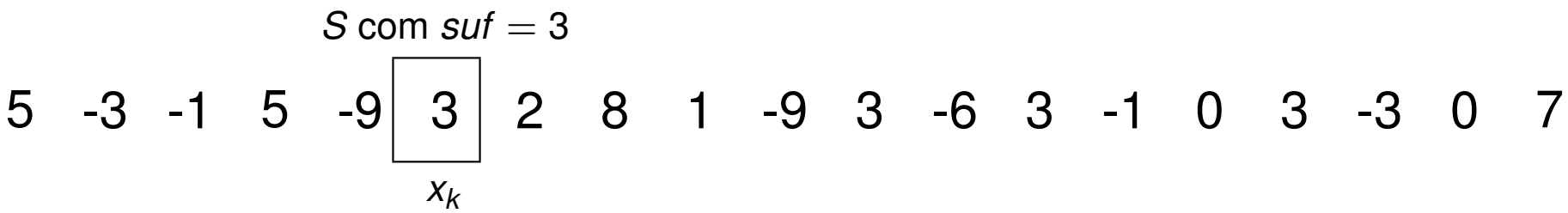
Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$

S vazio pois $suf < 0$

5	-3	-1	5	-9	3	2	8	1	-9	3	-6	3	-1	0	3	-3	0	7
				x_k														

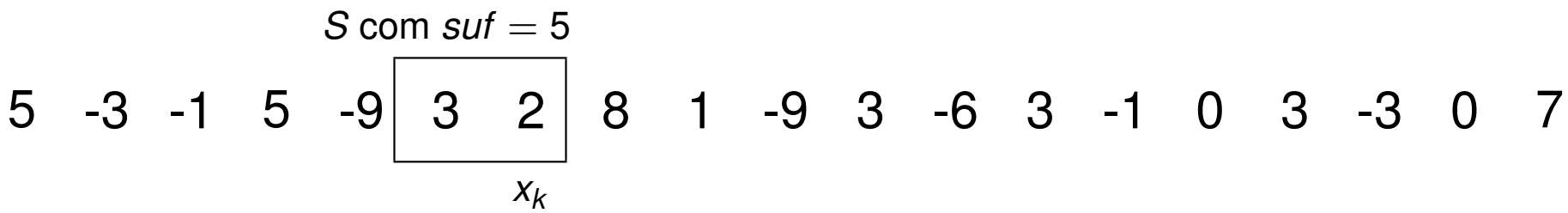
Exemplo de sufixo máximo

Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$



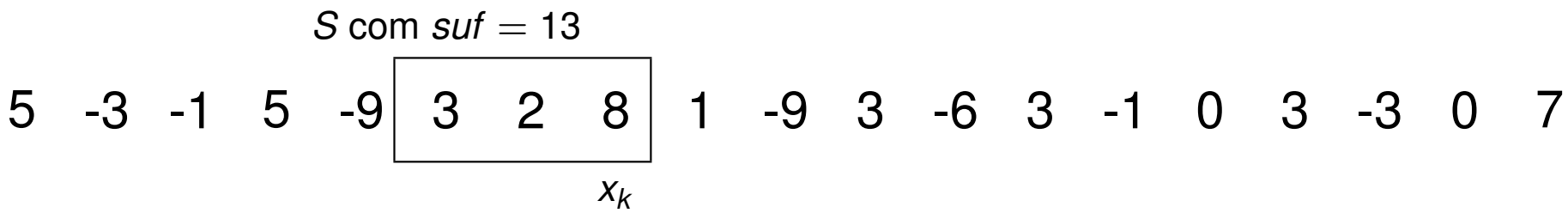
Exemplo de sufixo máximo

Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$



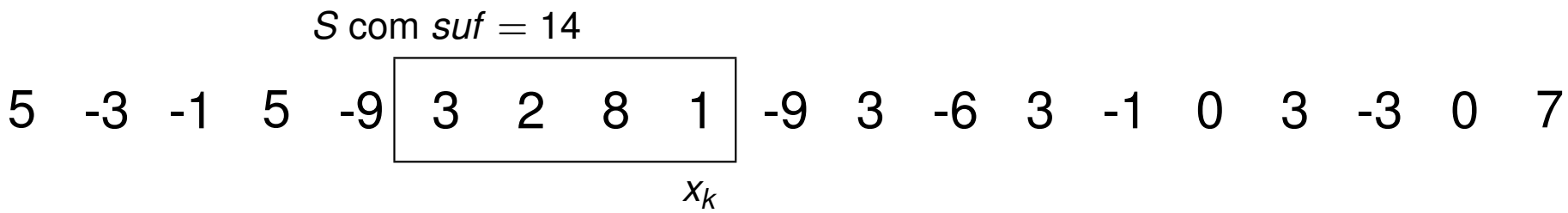
Exemplo de sufixo máximo

Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$



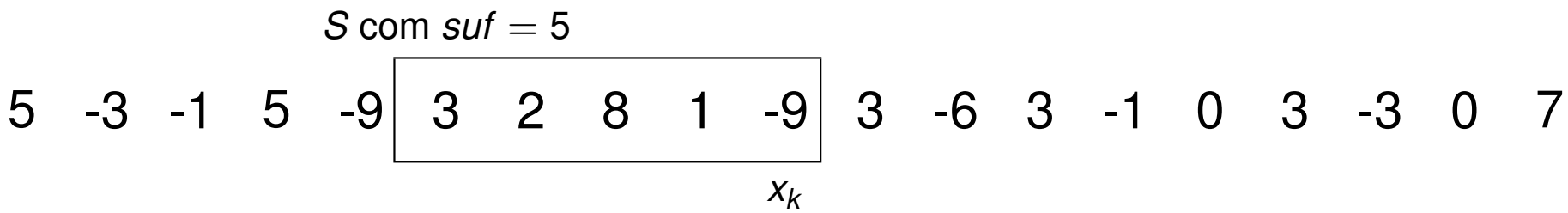
Exemplo de sufixo máximo

Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$



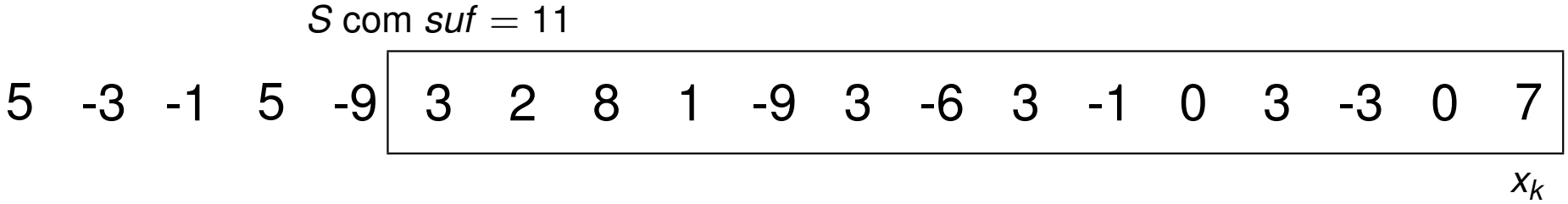
Exemplo de sufixo máximo

Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$



Exemplo de sufixo máximo

Sufixo máximo S da subsequência $x_1, x_2, \ldots, x_k$



Um algoritmo sequencial eficiente

- O algoritmo sequencial varre a sequência de n números da esquerda para a direita, desde x_1 até x_n.
- Suponha que de x_1 até x_k, a subsequência máxima M obtida tenha soma max.
- Vamos considerar o próximo número x_{k+1}.

M de soma $max = 6$ S com $suf = 5$

5	-3	-1	5	-9	3	2	8	1	-9	3	-6	3	-1	0	3	-3	0	7
						x_k	x_{k+1}											

Se x_k não pertence a M então temos que usar o sufixo máximo:

if $suf + x_{k+1} > max$
then adiciona x_{k+1} *a* S *e subsitui* M *por* S
else if $suf + x_{k+1} > 0$
then adiciona x_{k+1} *a* S
else reset S *para vazio*

Um algoritmo sequencial eficiente

- O algoritmo sequencial varre a sequência de n números da esquerda para a direita, desde x_1 até x_n.
- Suponha que de x_1 até x_k, a subsequência máxima M obtida tenha soma max.
- Vamos considerar o próximo número x_{k+1}.

M de soma $max = 6$ S com $suf = 5$

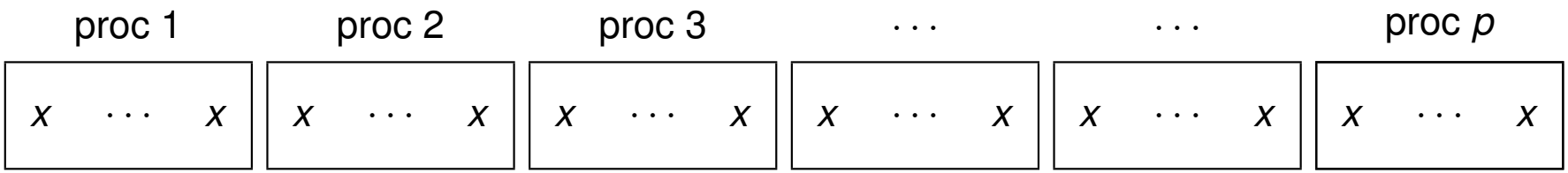
5	-3	-1	5	-9	3	2	8	1	-9	3	-6	3	-1	0	3	-3	0	7
					x_k	x_{k+1}												

Se x_k não pertence a M então temos que usar o sufixo máximo:

if $suf + x_{k+1} > max$
then adiciona x_{k+1} *a* S *e subsitui* M *por* S
else if $suf + x_{k+1} > 0$
then adiciona x_{k+1} *a* S
else reset S *para vazio*

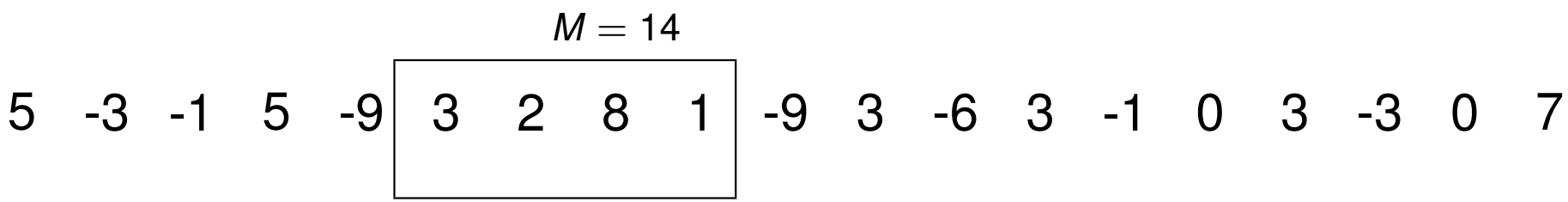
Como projetar o algoritmo paralelo

- Algoritmo paralelo de C. E. R. Alves, E. N. Cáceres e S. W. Song.
- Particionamos os n números dados entre os p processadores, cada um recebendo $1/p$ dos números, ou seja, n/p números.
- Cada processador resolve o problema de subsequência máxima usando os n/p números atribuídos a ele.
- Problema: e depois? Como prosseguir?



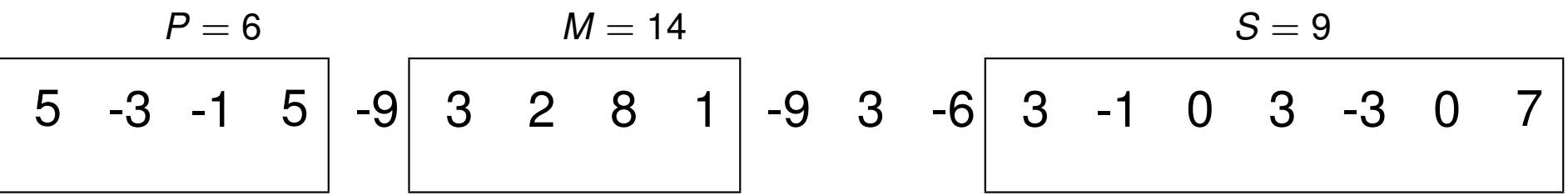
Subseq. máxima divide os números em duas partes

- Em cada processador, primeiro obtemos a subsequência máxima M.
- M vai dividir os demais números no processador em duas partes: uma à esquerda de M e outra à direita de M.
- Obs: as partes esquerda e direita podem ser vazias.



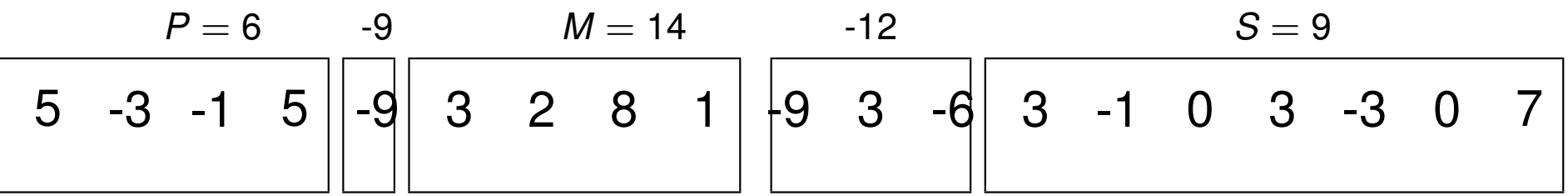
Subseq. máxima divide os números em duas partes

- Na parte direita, vamos obter o sufixo máximo S.
- Na parte esquerda, vamos obter o prefixo máximo P, calculado de maneira análoga ao sufixo máximo.
- O trecho entre P e M tem soma negativa. Do mesmo modo o trecho entre S e M tem soma negativa.



Cada processador calcula 5 números

- Na parte direita, vamos obter o sufixo máximo S.
- Na parte esquerda, vamos obter o prefixo máximo P, calculado de maneira análoga ao sufixo máximo.
- O trecho entre P e M tem soma negativa. Do mesmo modo o trecho entre S e M tem soma negativa.



Podemos resumir os dados em apenas no máx. 5 números, como acima.

Fase de combinar os resultados

- Uma vez os dados de cada processador podem ser condensados em apenas no máximo 5 números (juntamente com os intervalos associados), fica fácil combinar os resultados obtidos nos processadores.
- Todos os processadores enviam os 5 números a um mesmo processador, digamos processador 1.
- Processador 1 recebe um total de $5p$ números e executa o algoritmo sequencial para obter a resposta final.

Gráfico Tempo $\times$ Número de processadores

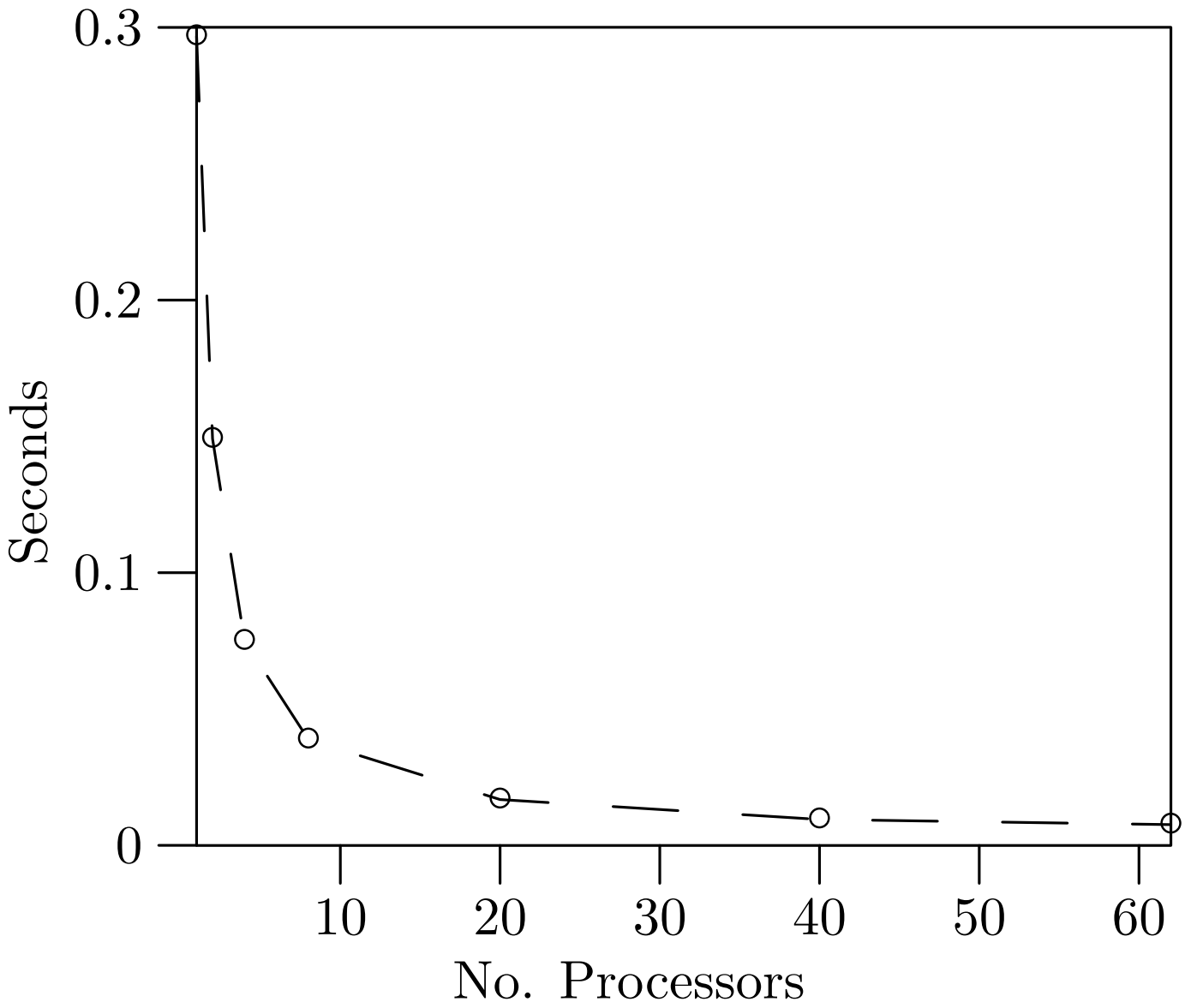
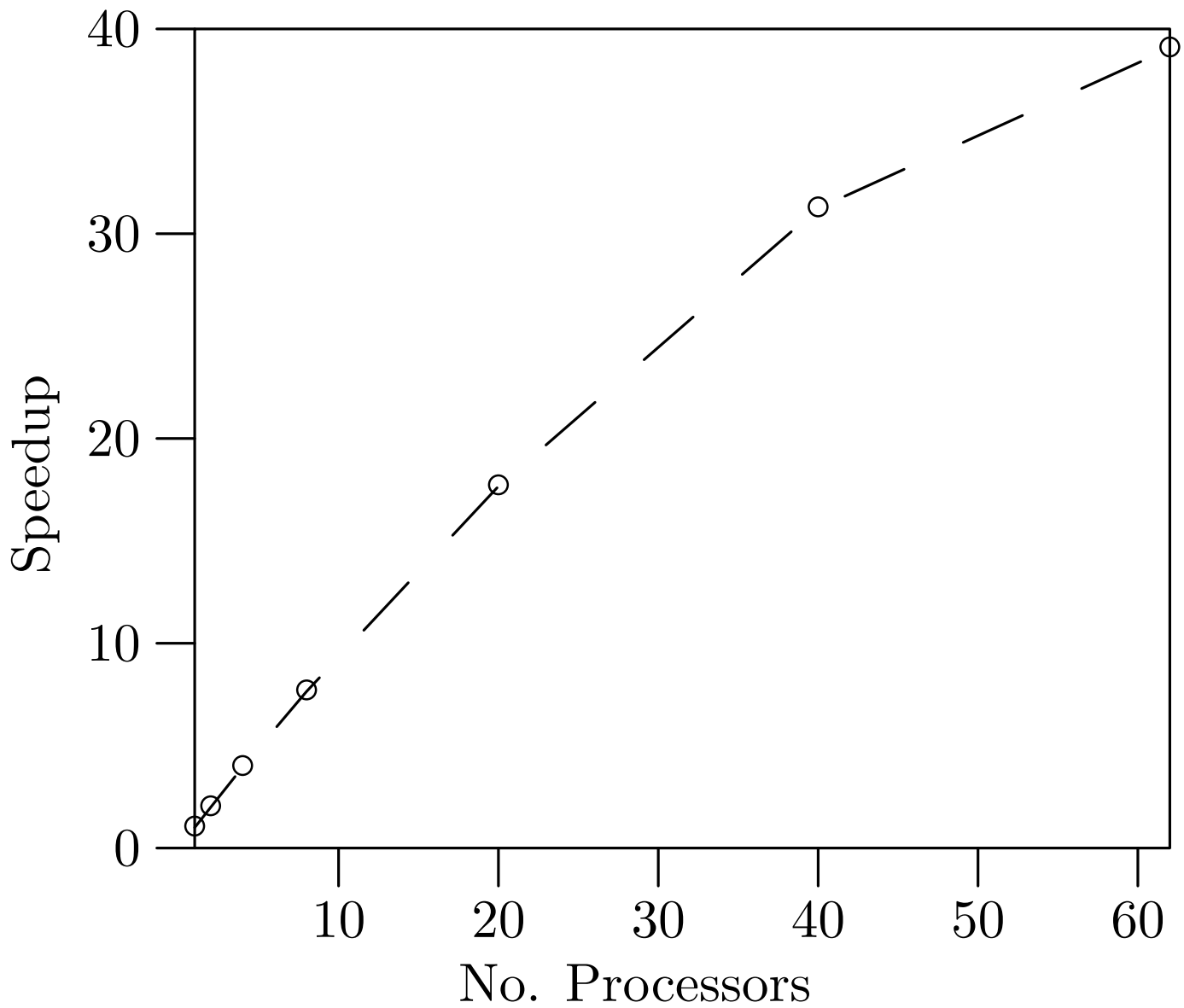


Gráfico Speedup $\times$ Número de processadores



O problema de todas as subsequências maximais

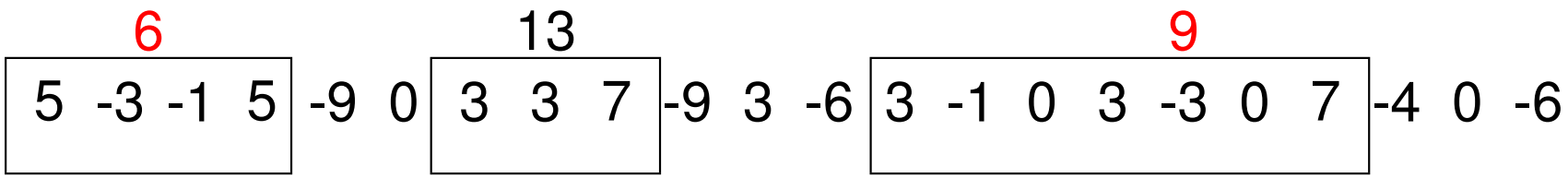
- É uma extensão do problema básico já visto.
- Depois de obter a subsequência máxima, obtenha também as subsequências maximais das partes restantes da sequência original, recursivamente. No final, apenas sobram números negativos.

Exemplo: Dada a sequência

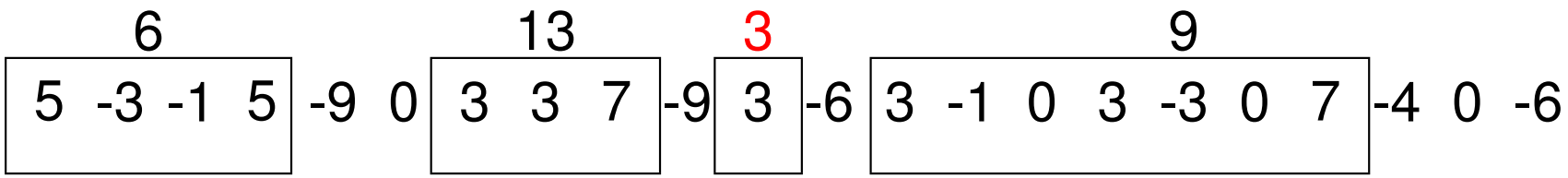
5 -3 -1 5 -9 0 3 3 7 -9 3 -6 3 -1 0 3 -3 0 7 -4 0 -6

Obtenha a subsequência máxima

Obtenha as subsequências maximais restantes



E assim por diante ...



Algoritmo sequencial e algoritmo paralelo

- Ruzzo e Tompa (1999) apresentam um algoritmo de tempo $O(n)$.
- Não é trivial obter um algoritmo paralelo que requer apenas $O(1)$ rodadas de comunicação.
- Cada processador resolve o problema de todas subsequências maximais não vai resolver.
- Cada processador pode produzir muitas subsequências maximais.
- Precisamos caracterizar em que condições uma subsequência maximal local pode ser uma candidata para ser concatenada a outras subsequências maximais em outros processadores.
- Alves, Cáceres e Song apresentaram um algoritmo paralelo que requer $O(1)$ rodadas de comunicação.

Conclusão

- Projetar um algoritmo paralelo frequentemente requer grande esforço.
- Tentar obter bons speedups para problemas que possuem algoritmos sequenciais eficientes é particularmente desafiante.
- O problema de subsequência máxima possui um algoritmo sequencial elegante de tempo $O(n)$.
- Mostramos como obter uma solução paralela que requer um número constante de rodadas de comunicação.

FIM

E assim concluo a palestra.

Obrigado!

O Desafio do Projeto de Algoritmos Paralelos

Siang Wun Song
Universidade Federal do ABC (UFABC) e Universidade de São Paulo (USP)

ERAD-SP 2011

São José dos Campos - 27 a 29 de julho de 2011