Algoritmo CGM para o Problema de Seleção

Baseado em artigos de Einar Saukas e Siang Song

O Problema da Seleção

Considere um conjunto de n elementos.

Dado um inteiro k ($1 \leq k \leq n$) deseja-se selecionar o k-ésimo menor elemento do conjunto.

Casos particulares:

$k = 1$: obter o mínimo

$k = n$: obter o máximo

$k = \lceil n/2 \rceil$: obter a mediana

Trabalhos Anteriores

O problema de seleção pode ser resolvido em tempo linear no modelo sequencial.

No modelo PRAM o problema é extensivamente estudado.

Há poucos trabalhos no modelo de memória distribuída:

> [Santoro, Sidney e Sidney 1992]: algoritmo probabilístico usando O($\log\log k + \log p$) rodadas de comunicação, onde p é o no. de processadores.
>
> [Bader e JáJá 1996]: algoritmo determinístico usando $O(\log(n/p^2))$ rodadas de comunicação.

Apresentamos um algoritmo determinístico que requer $O(\log p)$ rodadas de comunicação.

Técnica de Redução do Tamanho do Problema

- Procurar reduzir a entrada de n elementos para n/p, que cabem então num único processador. Pode-se então usar um algoritmo sequencial de seleção, em tempo $O(n/p)$.

Mostramos como a entrada pode ser reduzida de n para n/p em $O(\log p)$ rodadas de comunicação.

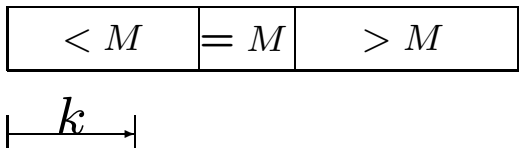
Como Reduzir o Tamanho do Problema

Cada processador calcula a mediana de seus dados (inicialmente n/p números).

Cada processador envia a mediana para o processador 1.

O processador 1 calcula a mediana M das p medianas. Prova-se que M tem *rank* entre $n/4$ e $3n/4$.

Sabendo-se a quantidade total de números $< M$, $= M$ e $> M$, pode-se descartar números que não podem ser resposta procurada:



No exemplo, o número procurado deve estar entre os números $< M$. Os $= M$ e $> M$ podem ser descartados. Temos:

Quantidade de números que sobram $\leq 3n/4$.

Uso da Mediana Ponderada das Medianas

Cada tamanho original N é reduzida a no máximo $3N/4$ em um passo de redução. Em $O(\log p)$ passos, reduziremos os n inciais para $O(n/p)$.

Importante: A quantidade de números que sobram em cada processador pode variar. Uma redistribuição de dados seria necessária em cada passo.

Para evitar essa redistrição, usamos a *mediana ponderada* M' das medianas calculdas em cada processador.

Dados p números distintos $m_1, m_2, \ldots, m_p$ com corresp. pesos positivos $w_1, w_2, \ldots, w_p$ tais que $\sum_{1 \leq i \leq p} w_i = 1$, a mediana ponderada é o elemento m_k que satisfaz $\sum_{i, m_i < m_k} w_i \leq \frac{1}{2}$ e $\sum_{i, m_i > m_k} w_i \leq \frac{1}{2}$.

Prova-se que o rank de M' também está entre $n/4$ e $3n/4$, portanto teremos o mesmo efeito desejado.

Algoritmo CGM proposto

Algorithm 1 Select the kth smallest element.

Input: Set A of n elements distributed among the p processors, each processor i with $n_i = n/p$ elements, and an integer k, $1 \leq k \leq n$.

Output: An element a_i of A such that *rank*$(a_i, A) = k$.

(1) Set $N := n$

(2) Repeat until $N \leq n/(cp)$

(2.1) Each processor i computes the median m_i of its n_i elements

(2.2) Each processor i sends m_i and n_i to processor 1

(2.3) Processor 1 computes the weighted median M

(2.4) Processor 1 broadcasts M to all other processors

(2.5) Each processor i computes l_i, e_i, g_i, respectively the numbers of its local elements less than, equal to, or greater than M

(2.6) Each processor i sends l_i, e_i, g_i to processor 1

(2.7) Processor 1 computes $L = \sum_{1 \leq i \leq p} l_i$, $E = \sum_{1 \leq i \leq p} e_i$, $G = \sum_{1 \leq i \leq p} g_i$, respectively the total numbers of elements less than, equal to, or greater than M

(2.8) Processor 1 broadcasts L, E, G to all other processors

(2.9) One of the following:

if $L < k \leq L + E$ then return solution M and stop

if $k \leq L$ then each processor i discards all but those elements less than M and set $N := L$

if $k > L + E$ then each processor i discards all but those elements greater than M and set $N := G$ and $k := k - (L + E)$

(3) All the remaining N elements are sent to processor 1

(4) Processor 1 solves the remaining problem sequentially

— End of Algorithm —

Parâmetro c

Cada rodada de comunicação, exceto a última, envolve a transmissão de $O(p)$ números.

A última rodada é a mais custosa pois todos os números que sobram devem ser transmitidos.

Note que o comando Repeat until $N \leq n/(cp)$.

O parâmetro c pode ser escolhido para obter o melhor tempo de execução numa dada máquina. O seu efeito é o seguinte:

O número de rodadas de comunicação é $O(\log cp)$.

A quantidade de dados transmitidos na última rodada: $O(n/(cp))$.

Portanto: maior o valor de c implica mais rodadas de comunicação mas menos dados transmitidos na última rodada.

Implementação

Parsytec PowerXplorer de 16 nós cada nó com:

. um processador PowerPC601 e um processador de comunicação T805

. 32 Mbytes de memória local

Usamos PVM e interface nativa Parix (dando resultados semelhantes).

Testamos com dois tipos de entradas:

Entrada uniforme: os dados de entrada são alocados nos processadores de maneira aleatória.

Entrada pior caso: os dados são alocados nos processadores de forma a produzir o pior desbalanceamento de carga.

Nessa máquina, após experimentar vários valores, achamos que o melhor é adotar o valor $c = 20$.

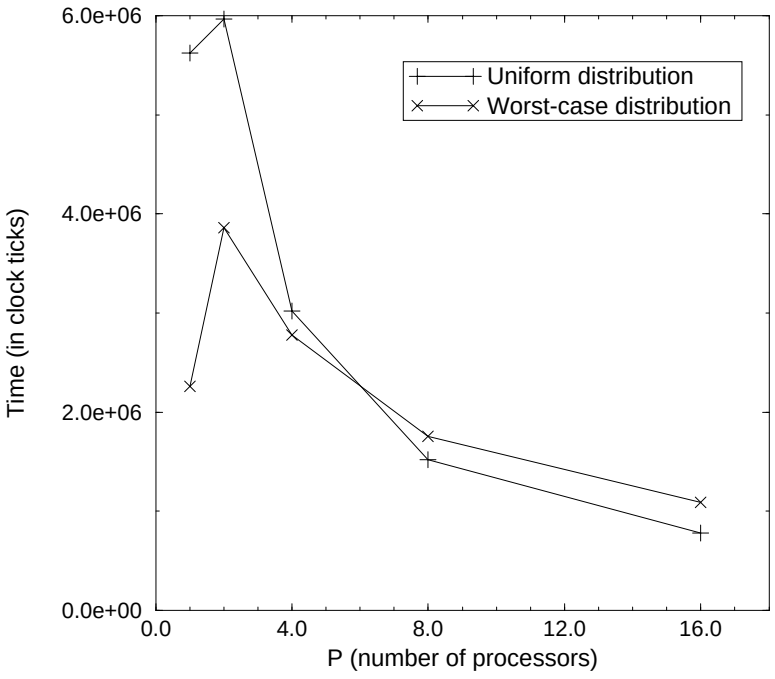
Resultados

Tempos de execução (em micro-segundos):

n	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$
1024	8727	5767	3372	3080	4714
2048	18896	11665	5887	3885	4794
4096	39032	23901	11730	6457	5903
8192	78618	47190	23654	12358	8077
16384	155944	95979	46693	24191	14005
32768	308148	189749	93867	47397	25552
65536	613099	375037	185122	94915	48508
131072	1223813	746283	365247	187116	95927
262144	2429358	1496103	727908	369437	187960
524288	4840124	2968610	1457379	736184	369990

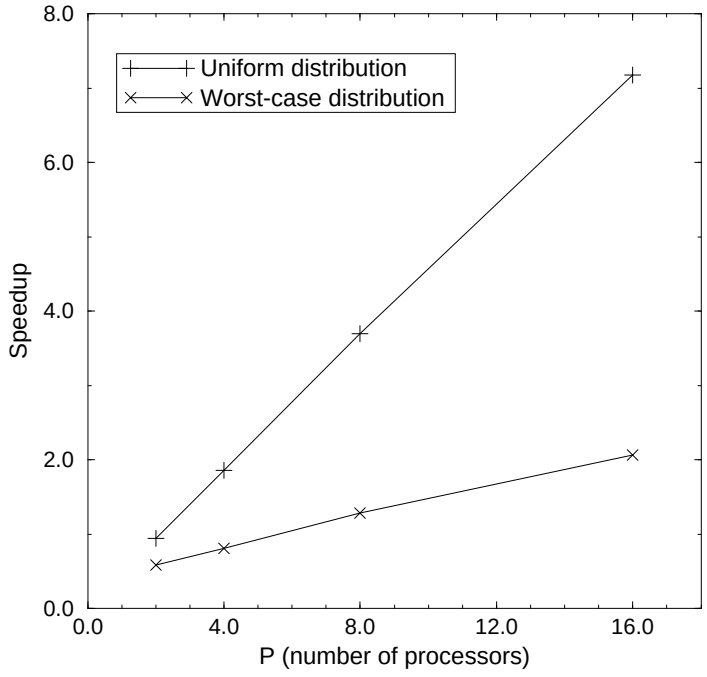
Curvas dos Tempos

Algorithm 1



Curvas dos Speedups

Algorithm 1



Conclusão

• Houve uma preocupação de minimizar a quantidade de dados transmitidos em cada rodada de comunicação.

• Introduzimos um parâmetro c que pode ser calibrado experimentalmente.