Multiplicação de Matriz-Vetor

Considere a multiplicação de uma matriz ***A*** densa (***n x n***), com um vetor ***x*** (***n x 1***) para produzir um vetor ***y*** (***n x 1***) como resultado. O pseudocódigo abaixo mostra um algoritmo sequencial para o problema (Grama et al. 2003):

```
procedure MAT_VECT ( A, x, y)
begin
   for i := 0 to n - 1 do
   begin
      y[ i ]:=0;
      for j := 0 to n - 1 do
          y[ i ] := y[ i ] + A[ i, j ] * x[ j ];
   endfor;
end MAT_VECT
```

Faça um projeto de algoritmo paralelo seguindo a metodologia PCAM para o problema acima, usando o particionamento 2-D de bloco de dados. O particionamento 2-D de bloco de dados particiona a matriz ***A*** tanto em linhas quanto em colunas (além do vetor ***x***). Em outras palavras, cada tarefa terá um bloco/quadrante (***n/sqrt(T) x n/sqrt(T)***) da matriz ***A*** e ***n / sqrt(T)*** elementos do vetor ***x***, onde ***T*** é o número de tarefas. Exemplificando, se houver ***n²*** tarefas, cada uma possuirá exatamente ***1*** elemento da matriz ***A*** e um elemento do vetor ***x***.

Descreva, explicitamente, de forma textual e gráfica (se necessário), o particionamento, a comunicação, a aglomeração e o mapeamento, considerando ao final o uso de ***P*** processos e ***PROC*** elementos de processamento em um cluster de computadores.

Considere que este algoritmo paralelo deve executar o mais rápido possível.
Por último, considere que forneceremos este projeto que vocês estão fazendo agora, para outro grupo implementá-lo remotamente. Espera-se que o seu projeto seja suficientemente detalhado, dentro do tempo disponível, para que a outra equipe possa fazer a implementação adequadamente.

Resposta:

Particionamento (versão com particionamento 2D da matriz A):

A matriz de ordem N é particionada em duas dimensões (2D), usando N^2 tarefas. Cada tarefa possui um elemento de A[i,j] e o respectivo elemento do vetor X[j] correspondente à coluna de A[x,j] que a tarefa está computando. Após as multiplicações, as quais podem ser feitas em paralelo, as tarefas responsáveis por uma linha i de A fazem as suas somas das multiplicações realizadas, produzindo um respectivo elemento i do vetor Y. Tais somas podem ser feitas com uma operação de redução por linha, de ordem logarítmica.

Comunicação:

As comunicações entre as tarefas são responsáveis por fornecer os dados de entrada para tarefas e por recuperar os resultados das computações feitas nas tarefas. Desse modo, cada tarefa recebe o seu respectivo elemento de A e as tarefas responsáveis pelos dados da última coluna A também recebem os respectivos elementos do vetor X.

Ao iniciar a execução, as tarefas da última coluna repassam seus valores de X para a tarefa responsável pelo elemento da diagonal principal da respectiva linha. Em outras palavras, a tarefa responsável pelos dados da posição A[j,j] recebe o valor de X[j]. Esta tarefa da diagonal principal de A[j,j], por sua vez, faz um broadcast deste valor de X para todas as tarefas que computam dados desta coluna de A[*,j].
Após este broadcast, todas as N^2 tarefas têm o elementos de A e de X necessários para realizar, em paralelo, a multiplicação necessária (A[i,j]*X[j]).
Após a tarefa de multiplicar, cada tarefa contribui com a sua linha de tarefas em uma operação de redução, onde log N iterações/passos realizam em paralelo as somas necessárias para se obter um Y[i]. A cada iteração haverá N'/2 comunicações, onde N' = {N/2, N/4, N/8, …, 1) .

Aglomeração:

Dada a natureza 2D da especificação do problema e a plataforma alvo deste algoritmo (um cluster de computadores), sugere-se o agrupamento das N^2 tarefas em P processos, onde P equivale ao número de elementos de processamento (ou núcleos) disponíveis. Esta aglomeração permitirá aumentar a granularidade de cada processo e diminuir a comunicação necessária para atingir o objetivo final.
Desta forma, a aglomeração das tarefas será feita em duas dimensões, onde cada processo receberá um bloco de dados de A com duas dimensões, contendo N / (sqrt(P) x N / sqrt(P)) elementos. Por exemplo, caso a matriz A tenha 8x8 elementos e haja 4 (2x2) processos, cada processo será responsável por um bloco de dados de A, equivalente a uma submatriz de 4x4. Caso houvesse a mesma carga de trabalho e 64 processos, cada um dos 64 (8x8) blocos receberia um quadrante de 1x1 elemento, voltando ao particionamento original do problema em tarefas.
Os valores de X serão atribuídos aos blocos de processos responsáveis pelas últimas colunas de A, de maneira análoga à distribuição de X já feita no particionamento. Os (n / sqrt(p)) elementos de X necessários aos demais processos serão repassados por broadcast à diagonal principal daquela linha dentro do bloco.
Cada processo computará sequencialmente a sua sequência de somas das suas multiplicações, produzindo um somatório parcial das respectivas linhas de A.
Em um segundo passo, estes sqrt(P) processos responsáveis pelas mesmas linhas de A farão operações de redução considerando suas respectivas linhas, gerando assim novos valores finais para o vetor Y.

Mapeamento:

Considerando que os nós do cluster possuem um desempenho homogêneo, o mapeamento de P processos em PROC Elementos de Processamento ocorrerá por meio de uma fila circular (Round-Robin). Neste caso, se P == PROC, então cada Elemento de Processamento receberá exatamente um processo.
Caso o desempenho dos nós do cluster seja diferente, então o mapeamento dos nós deve considerar a diferença de desempenho entre os elementos de processamento, o que pode ser feito estaticamente (antes da execução) ou dinamicamente (em tempo de execução considerando a atribuição ao nó com a menor carga de trabalho em andamento). Se o escalonamento for dinâmico, o mapeamento de processos aos nós pode ser guiado por métricas como: número de processos na fila de pronto para execução, uso de CPU, quantidade de bytes trocados em swap de disco, entre outras.