An Introduction to Parallel Algorithms

By: Arash J. Z. Fard

GRSC7770 – Graduate Seminar

11/9/2009
Topics

1. Definition of Parallel Algorithm
2. Task/Channel Model
3. Foster’s Design Methodology
4. An Example
Reference

Definition of Parallel Algorithm

A parallel algorithm is an algorithm which can be executed a piece at a time on many different processing devices, and then put back together again at the end to get the correct result. (wikipedia)

A parallel program usually contains some tasks which are executed concurrently on different processors, and may communicate with each other.

The main purpose for designing and using a parallel algorithm is reducing total running time of a process
A task is a program, its local memory, and a collection of I/O ports.

Receive: Synchronous
Send: Asynchronous

A channel is a message queue that connects one task’s output port with another task’s input port.
Foster’s Design Methodology

Problem -> Partitioning -> Communication

Mapping -> Agglomeration
1. Partitioning

• **Domain Decomposition**

  Dividing *data into pieces* and then determine how to associate computations with data

• **Functional Decomposition**

  First dividing the *computation into pieces* and then determine how to associate data items
1.1. Domain Decomposition
1.2. Functional Decomposition

- Acquire patient images
- Register images
- Determine image locations
- Track position of instruments
- Display image
Partitioning Considerations

- An order of magnitude more primitive tasks than processors
- Minimized redundant computation and redundant data structure storage
- Roughly the same size for primitive tasks
- The number of tasks is an increasing function of the problem size
2. Communication

• Local Communication

A task communicating with a small number of other tasks in order to perform a computation

• Global Communication

A significant number of the primitive tasks must contribute data in order to perform a computation
Communication Considerations

- The communication operations are **balanced** among the tasks.
- Each task communicates with only a **small number of neighbors**.
- Task can perform their **communications concurrently**.
3. Agglomeration

Agglomeration is the process of grouping tasks into larger tasks in order to improve performance or simplify program.
Agglomeration Considerations

- Increased the locality of the parallel algorithm
- Replicated computation take less time than the communication they replace
- The amount of replicated data is small enough to allow the algorithm to scale
- Agglomerated tasks have similar computational and communications costs
- The number of tasks is as small as possible, but greater than number of processors
- The trade-off between the chosen agglomeration and the cost of modification
4. Mapping

**Goals:** Maximize Processor Utilization, Minimize Interprocessor Communication

**Note:** Increasing processor utilization and minimizing interprocessor communication are often conflicting goals
Mapping Considerations

- Designs based on one task per processors and multiple tasks per processor have been considered.
- Both static and dynamic allocation of tasks to processors have been evaluated.
- If a dynamic allocation of tasks to processors has been chosen, the manager is not a bottleneck to performance.
An Example: Multiply Two Matrices

\[
\begin{pmatrix}
a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n,1} & a_{n,2} & \cdots & a_{n,n}
\end{pmatrix}
\times
\begin{pmatrix}
b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\
b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
b_{n,1} & b_{n,2} & \cdots & b_{n,n}
\end{pmatrix}
\]

\[
A_{nn} \times B_{nn} = 
\begin{pmatrix}
a_{11}b_{11} & a_{12}b_{21} & a_{13}b_{31} & \cdots & a_{1n}b_{n1} \\
a_{21}b_{12} & a_{22}b_{22} & a_{23}b_{32} & \cdots & a_{2n}b_{n2} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
a_{n1}b_{1n} & a_{n2}b_{2n} & a_{n3}b_{3n} & \cdots & a_{nn}b_{nn}
\end{pmatrix}
\]
Multiply Two Matrices (cont.)
Multiply Two Matrices (cont.)

\[
\begin{array}{cccc}
  a_{11} & a_{12} & \ldots & a_{14} \\
  a_{21} & a_{22} & a_{23} & a_{24} \\
  a_{31} & a_{32} & a_{33} & a_{34} \\
  a_{41} & a_{42} & a_{43} & a_{44} \\
\end{array}
\quad
\begin{array}{cccc}
  b_{11} & b_{12} & b_{13} & b_{14} \\
  b_{21} & b_{22} & b_{23} & b_{24} \\
  b_{31} & b_{32} & b_{33} & b_{34} \\
  b_{41} & b_{42} & b_{43} & b_{44} \\
\end{array}
\]

\[
\begin{array}{cccc}
  a_{11} & a_{12} & a_{13} & a_{14} \\
  a_{21} & a_{22} & a_{23} & a_{24} \\
  a_{31} & a_{32} & a_{33} & a_{34} \\
  a_{41} & a_{42} & a_{43} & a_{44} \\
\end{array}
\quad
\begin{array}{cccc}
  b_{11} & b_{12} & b_{13} & b_{14} \\
  b_{21} & b_{22} & b_{23} & b_{24} \\
  b_{31} & b_{32} & b_{33} & b_{34} \\
  b_{41} & b_{42} & b_{43} & b_{44} \\
\end{array}
\]
Conclusion

Now You should be able to

1. Define Parallel Algorithm
2. Describe Task/Channel Model
3. Use Foster’s Design Methodology
   i. Partitioning
   ii. Communication
   iii. Agglomeration
   iv. Mapping
Question?
Write pseudo code of a parallel algorithm to multiply 2 matrices. Map it on a Quad-Processor system (SMP).

Due time: 11/16/2009, 5:00pm