ECE 669
Parallel Computer Architecture
Lecture 23
Parallel Compilation
Parallel Compilation

° Two approaches to compilation
  • Parallelize a program manually
  • Sequential code converted to parallel code

° Develop a parallel compiler
  • Intermediate form
  • Partitioning
    - Block based or loop based
  • Placement
  • Routing
Compilation technologies for parallel machines

Assumptions:

Input: Parallel program

Output: “Coarse” parallel program
& directives for:

• Which threads run in 1 task
• Where tasks are placed
• Where data is placed
• Which data elements go in each data chunk

Limitation: No special optimizations
for synchronization --
synchro mem refs treated
as any other comm.
Toy example

Loop parallelization

Adding a total of 4n integers, \(a_1, a_2, \ldots, a_{4n}\), on a 4-processor system.

Processor 0 will execute \(a_0 + a_2 + \cdots + a_{n-1}\).
Processor 0 will execute \(a_n + a_{n+2} + \cdots + a_{2n}\).
Processor 0 will execute \(a_{2n} + a_{2n+2} + \cdots + a\).
Processor 0 will execute \(a_{3n} + a_{3n+2} + \cdots + a\).
Example

- **Matrix multiply**

- **Typically,**

  \[
  \text{FORALL}\ i \\
  \text{FORALL}\ j \\
  \text{FOR}\ k \\
  C[i, j] = C[i, j] + A[i, k] \times B[k, j]
  \]

- **Looking to find parallelism...**
Choosing a program representation...

- **Dataflow graph**

  - No notion of storage
  - Data values flow along arcs
  - Nodes represent operations
Compiler representation

For certain kinds of structured programs

\[ A[i, j] \]

Unstructured programs

Data array
Index expressions
LOOP nest

Data array

Communication weight

Task A
Task B
Process reference graph

FORALL $i$ FROM 0 to 5
\[ C[i] = A[i] + B[i] \]

- Nodes represent threads (processes) computation
- Edges represent communication (memory references)
- Can attach weights on edges to represent volume of communication
- Extension: precedence relations edges can be added too
- Can also try to represent multiple loop produced threads as one node

......Not very useful
• Allocate data items to nodes as well
• Nodes: Threads, data objects
• Edges: Communication
• Key: Works for both shared-memory, object-oriented, and dataflow systems! (Msg. passing)
PCG for Jacobi relaxation

Fine PCG

Coarse PCG

○ : Computation
○ : Data
Compilation with PCGs

Fine process communication graph

Partitioning

Coarse process communication graph
Compilation with PCGs

Fine process communication graph

Coarse process communication graph

Partitioning

Coarse process communication graph
... other phases, scheduling.
Dynamic?

MP:

Placement

ECE669 L23: Parallel Compilation
April 29, 2004
Parallel Compilation

- Consider loop partitioning
- Create small local compilation
- Consider static routing between tiles
- Short neighbor-to-neighbor communication
- Compiler orchestrated
Flow Compilation

- Modulo unrolling
- Partitioning
- Scheduling
Modulo Unrolling – Smart Memory

- Loop unrolling relies on dependencies
- Allow maximum parallelism
- Minimize communication

(a) Data

```
A[]
B[]
```

(b) Code

```
for(i=0;i<100;i++)
```

(c) Code

```
for(i=0;i<100;i+=4)
{
    A[i+1]=A[i+1]*B[i+2]
}
```
Array Partitioning – Smart Memory

- Assign each line to separate memory
- Consider exchange of data
- Approach is scalable

```c
i' = 0;
for(i=0; i<100; i+=4) {
    A_0[i'] = A_0[i'] * B_0[i']
    A_1[i'] = A_1[i'] * B_1[i']
    A_2[i'] = A_2[i'] * B_2[i']
    A_3[i'] = A_3[i'] * B_3[i']
    A_0[i'] = A_0[i'] * B_0[i' + 1]
    i' = i' + 1
}
```
Communication Scheduling – Smart Memory

- Determine where data should be sent
- Determine when data should be sent
Speedup for Jacobi – Smart Memory

- Virtual wires indicates scheduled paths
- Hard wires are dedicated paths
- Hard wires require more wiring resources
- RAW is a parallel processor from MIT

![Speedup scalability for jacobi](chart)

- Custom-hard-wires
- Custom-virtual-wires
- Raw

Ntiles

Speedup

0  4  8  12  16

0  2  4  6  8  10  12  14  16
Partitioning

- Use heuristic for unstructured programs
- For structured programs...

...start from:

Arrays
A  B  C

List of arrays
L0  L1  L2

List of loops

Loop Nests
Notion of Iteration space, data space

E.g.

\[
\begin{align*}
\text{Forall } i \\
\text{Forall } j \\
\end{align*}
\]

- A Matrix
- Data space
- Iteration space
- Represents a “thread” with a given value of \( i, j \)
Notion of Iteration space, data space

E.g.
For all $i$
For all $j$


- **Partitioning**: How to “tile” iteration for MIMD M/Cs data spaces?
Loop partitioning for caches

- **Machine model**

  - Assume all data is in memory
  - Minimize first-time cache fetches
  - Ignore secondary effects such as invalidations due to writes
Summary

- Parallel compilation often targets block based and loop based parallelism
- Compilation steps address identification of parallelism and representations
  - Graphs often useful to represent program dependencies
- For static scheduling both computation and communication can be represented
- Data positioning is an important for computation