Characterizing and Reducing Overhead in Parallel Applications

Shih-Hao Hung
hungsh@umich.edu

Prof. Edward S. Davidson
davidson@eecs.umich.edu

ARC Annual Conference
June 4, 1997

Advanced Computer Architecture Laboratory
Electrical Engineering and Computer Science Dept.
The University of Michigan, Ann Arbor
Outline

- Introduction
- Our approach
- Machine-application performance bound models
- CXbound
- Case study
- Conclusion
Introduction

- Parallel computers: peak performance vs. delivered performance

- Problems?

  Application

  Limited parallelism
  problem size
  data dependencies

  Algorithm

  Inefficient parallel algorithms
  lack of experience

  Parallel Program

  Parallel execution overhead
  communication
  synchronization

  Hardware

  Poor machine resource utilization
  inefficient scheduling
  cache misses
  network contention
  inefficient communication traffic, etc.
Our Approach

- Current compilers are far from ideal.
- Friendly and effective performance tools are highly desired.
- Future compilers.
Machine-Application Performance Bounds

System constraints  Gaps  Execution time

Behaviors not modeled

Dynamic load behavior

Multiple program phases

Overall load imbalance

Interprocessor communication

Partial parallelization

Finite cache effect

Data dependency, branches, pipeline bubbles

Compiler-inserted instructions

Mismatched application workload

Machine peak performance

Actual run time

IPCLM-Dynamic (IPCLMD) bound

IPCL-Multiphase (IPCLM) bound

IPC-Load balance (IPCL) bound

IP-Communication (IPC) bound

I-Parallelization (IP) bound

Ideal-parallel (I) bound

MACS-Cache (MACS$) bound

MAC-Schedule (MACS) bound

MA-Compiler (MAC) bound

M-Application (MA) bound

Machine (M) bound

Shih-Hao Hung  Characterizing and Reducing Overhead in Parallel Applications  Edward S. Davidson
Machine-Application Performance Bounds

• Previous work
  – Davidson et al., Uniprocessor bound models for Cray, ZS-1, Convex C-240, IBM RS/6000, KSR1, DEC Alpha, HP-PA., etc. ‘88-‘94
  – Boyd (Davidson), Bound models for shared-memory and load balance on the KSR1, ‘95

• New Development
  – Parallelization bound: Evaluating the effect of degree of parallelization,
  – Multiphase bound: Distinguishing temporal load imbalance and overall load imbalance
  – Multiple program regions.
  – Shared-memory or message-passing programs.
  – Automatic tool (CXbound) on the Convex Exemplar.
  – New bounding mechanisms using profiles.
CXbound

Parallelization
Communication
Operating System

Bound = \text{Serial} + \frac{\text{Para Load}}{N}

Load Balance Bound = \text{Serial} + \max_p \{\text{Para Load}_p\}

Multiphase Bound = \text{Serial} + \sum_{p,r} \max_{p,r} \{\text{Para Load}_{p,r}\}

*approximate computation workload by 1-proc. run

Workload reported by CXpa

CPU time
communications
I/O, OS interference

Computation

Time
serial
perfectly balanced parallel region
load imbalance
temporal load imbalance

Working, idle, spawn/join, barrier synchronization

Parallel Load Bound = \text{Serial} + \max_p \{\text{Para Load}_p\}
Case Study - Matrix Multiplication

```
program MM1
    integer iter
    do iter = 1, 100
      call MULT1
    end do
end

subroutine MULT1
    parameter (N=512)
    double precision a(N,N), b(N,N), c(N,N)
    integer i, j, k

    c initialization loop
    do i=1, N
      do j=1, N
        a(i,j) = sin(i*j*0.1)
        b(i,j) = cos(i*j*0.1)
      end do
    end do

    c multiplication loop
    do i=1, N
      do j=1, N
        c(i,j) = 0.0
        do k=1, N
          c(i,j) = c(i,j) + a(i,k) * b(k,j)
        end do
      end do
    end do
end

program MM2
    do j=1, N
      do i=1, N
        c(i,j) = 0.0
        do k=1, j
          c(i,j) = c(i,j) + a(i,k) * b(k,j)
        end do
      end do
    end do
end

program MM_LU
    do j=1, N
      do i=1, N
        c(i,j) = 0.0
        do k=1, j
          c(i,j) = c(i,j) + a(i,k) * b(k,j)
        end do
        do k=j+1, N
          c(i,j) = c(i,j) + a(i,k) * b(k,j)
        end do
      end do
    end do
```

Shih-Hao Hung

Edward S. Davidson
Case Study - MM1

Original code
- Characterization indicates communication overhead
Case Study - MM2

Loop #1 in the original code is not interchanged!
after interchanging indices of para loop #1 ...
Case Study - MM_LU

- overall load is balanced
- multiple phase load imbalance
Case Study - Finite Element Application

- Shared memory code
- 10+ parallel loops
- Good speedup within 1 hypernode

Problems:
- Imperfect (90%) parallelization
- High cost for interhypernode communication
- Difficult to balance the load for larger systems

Suggested Solutions:
- Improving degree of parallelization
- Better domain decomposition
- Relaxing scheduling constraints
Conclusions

- Performance characterization and tuning are essential to achieve high performance on parallel machines.
- Bound analysis can provide performance characterization that is useful for understanding application/machine performance behavior and for performance tuning.
- We show how performance bounds can be derived on the Convex Exemplar. In our case study, CXbound provides friendly, systematic, and effective performance characterization.