What SMT can do for You

SMT=Simultaneous Multithreading

John Hague, IBM Consultant
Nov 06
ECMWF (European Centre for Medium Range Weather Forecasts)

Growth in HPC performance


- Times 5 every 4 years
- Moore’s Law
The Problem (i.e. Challenge)

• For Parallel Programs using large number of Processors:
  – CPU performance increases much faster than Memory Access rates
  – Floating Point pipeline length increases (for scalar processors)
  – Increased number of CPUs produces more inter-CPU Communications

• Percentage of peak TFLOPS decreases
  – Currently about 11% for ECMWF 10 Day Forecast (without SMT)

• Need to make better use of CPU to overcome wait for
  – Floating Point Pipes
  – Memory Access
## CPI analysis for IFS (RAPS9) on Power5

**CPI = Cycles Per Instruction**

**IFS = ECMWF’s Integrated Forecast System**

<table>
<thead>
<tr>
<th>MAIN</th>
<th>T</th>
<th>Cycles</th>
<th>1,049,275,053,531</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Groups</td>
<td>371,422,635,466</td>
<td>0.354</td>
</tr>
<tr>
<td>B</td>
<td>GCT</td>
<td>16,540,437,851</td>
<td>0.016</td>
</tr>
<tr>
<td>C</td>
<td>Stalls</td>
<td>661,311,980,214</td>
<td>0.630</td>
</tr>
</tbody>
</table>

|               | C1   | LSU                  | 194,682,702,732    | 0.186 |
|               | C2   | FXU                  | 65,549,096,175     | 0.062 |
|               | C3   | FPU                  | 340,351,526,592    | 0.324 |
|               | C4   | Other                | 60,728,654,715     | 0.058 |

Thanks to Lawrence Hannon, IBM Austin, for assistance
# CPI analysis for IFS (RAPS9) on Power5

## LAITLI

<table>
<thead>
<tr>
<th>Component</th>
<th>Cycles</th>
<th>Groups</th>
<th>GCT</th>
<th>Stalls</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>18,544,481,257</td>
<td>4,812,590,956</td>
<td>77,943,390</td>
<td>13,653,946,911</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>. . .</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## CLOUDSC-3

<table>
<thead>
<tr>
<th>Component</th>
<th>Cycles</th>
<th>Groups</th>
<th>GCT</th>
<th>Stalls</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>18,536,291,356</td>
<td>7,638,718,046</td>
<td>639,681,301</td>
<td>10,257,892,009</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>. . .</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Detailed Breakdown

<table>
<thead>
<tr>
<th>Component</th>
<th>Cycles</th>
<th>Groups</th>
<th>GCT</th>
<th>Stalls</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>. . .</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SMT (Simultaneous Multi Threading)

• Make better use of Execution Units (particularly Floating Point pipes)
  – Two threads run on one physical PE
  – 2nd thread can be dispatched (at hardware level) while 1st thread waiting (e.g. for FP pipe or Load instruction)
  – No context switching overhead

• Upside
  – Up to 2x Performance improvement

• Downside
  – May get Cache Thrashing (unless twice as much Cache available)
  – May get Paging (unless twice as much Memory available)
SMT: Simplified Power5 Schematic

Ref: IEEE Micro; March-April 2004; Kalla, Sinharoy & Tendler
IBM Power5 Chip, A Dual-Core Multi-threaded Processor
SMT Operation

• **Initial Operation**
  – Two separate Program Counters (for 2 threads)
  – Shared Instruction Cache
  – Instruction Fetches alternate between threads
  – Up to 8 instructions from same thread per cycle into instruction buffer

• **Thread Select**
  – Selects 5 instructions from same thread to form a group for Issue Queues
  – Takes one entry in shared 20 slot Global Completion Table (GCT)

• **Issue Queues (shared)**
  – Allocate shared rename registers (120 FP & 120 GP)

• **Execution Units**
  – Up to 8 instructions can be executed, out of order, every cycle
  – Groups complete in order for each thread (one from each thread per cycle)
Enhanced SMT

• Dynamic Resource Balancing
  – Aims to prevent one thread hogging Issue Queues
  – Throttles thread if too many L2 misses or GCT entries by:
    • Reducing thread’s priority
    • Inhibiting thread’s instruction decoding
    • Flushing thread’s instructions waiting for dispatch

• Adjustable Thread Priority
  – 8 software controlled priority levels
  – Implemented by controlling instruction decoding cycles
  – Priority decreased if thread in
    • Idle loop Waiting for work
    • Spin loop waiting for lock
  – Priority Increased for
    • real time task
SMT on P5+ (p575 node)

- Node has 16 physical PEs = 8 dual-core chips
- AIX “sees” 32 PEs, and allocates two threads physical PE
- Parallel program can use MPI or OpenMP to double number of threads
  - Try not to double memory requirements
    - best to use OpenMP?
  - Needs appropriate binding of threads to PEs
- Programs should benefit from SMT if
  - PE pipes not fully utilised
    - Matrix multiply fully uses pipes
  - Memory bandwidth not fully utilised
  - Program is scaleable
## SMT benefit (on 16 CPU P5+)

<table>
<thead>
<tr>
<th>Code</th>
<th>FP pipe use</th>
<th>Memory access</th>
<th>GFLOP no SMT (16 thrds)</th>
<th>GFLOP SMT (32 thrds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>one program, multiple threads</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>requires bit reproducibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1 = s1 + 1.d0</td>
<td>One</td>
<td>None</td>
<td>5.0</td>
<td>10.0</td>
</tr>
<tr>
<td>s1 = s1 + 1.d0</td>
<td>Both</td>
<td>None</td>
<td>10.1</td>
<td>19.9</td>
</tr>
<tr>
<td>s2 = s2 + 1.d0</td>
<td>Both</td>
<td>None</td>
<td>50.2</td>
<td>60.3</td>
</tr>
<tr>
<td>s1 = s1 + 1.d0</td>
<td>Both</td>
<td>None</td>
<td>11.3</td>
<td>16.2</td>
</tr>
<tr>
<td>s10 = s10 + 1.d0</td>
<td>Both</td>
<td>Streamin g</td>
<td>3.0</td>
<td>3.3</td>
</tr>
</tbody>
</table>

```fortran
!*OMP PARALLEL DO
PRIVATE(S)
DO J=1,NSTRM
  DO I=1,N
    S=S+1
  ENDDO
WRITE(6,*) S
ENDDO
```
SMT for Parallel jobs (on P5+)

- If one copy of program takes T
  
  ![One Copy Diagram]

- If SMT factor is S, 2 copies of program take 2*T/S
  
  ![Two Copies Diagram]

- If scalability factor for doubling threads if f, program takes 2*T/f/S
  
  ![Double Threads Diagram]

- For IFS T799 on 1200 CPU, S=1.35, f=1.8
  - Speedup due to SMT is S*f/2 = 1.35*1.8/2 = 1.22
Binding (for P5+ p575)

- Use MEMORY_AFFINITY=MCM
  - Allocates memory to same resource (i.e. chip or 2 PEs on dual core p575) that as thread is running on

- Binding keeps all threads on specified PE
  - Use simple binding code

- If SMT enabled AIX “sees” 32 PEs per node

- To run without SMT, schedule 16 threads per node, and specify binding map
  - 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30

- To use SMT, schedule 32 threads per node, and specify binding map
  - 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

- To SMT when OpenMP may not use all threads, specify binding map (for 16 threads per task)
  - 0 2 4 6 8 10 12 14 1 3 5 7 9 11 13 15 16 18 20 22 24 26 28 30 17 19 21 23 25 27 29 31
ECMWF’s 4D-Var 2nd minimisation (RAPS8) communication time increased with run number

Each run about 40 min; 2nd minimisation about 12 min
Monitoring Effect

1 sec delay per node for monitoring

Thanks to Oliver Treiber (ECMWF) for this

Reboot
Explanation of monitoring effect

Initially

Task 0

Task 1

Task 2

Task 3

Comms

CPU

dt

dt

dt

dt

Every timestep: delay dt in CPU

After several hours

Every timestep (for N tasks):
 delay dt in CPU
 delay N*dt in Comms
IFS 4D-Var 2nd minimisation: SMT and Binding v threads (RAPS8)

384 PEs, 24 Nodes of P5+, 16 PEs each

Time

Threads per MPI Task for noSMT, (Threads per MPI Task)/2 for SMT

- No SMT, no bind
- SMT, no Bind
- No SMT, Bind
- SMT, Bind
UKMO Unified Model: SMT and Binding v threads

128 PEs, 8 Nodes of P5+, 16 PEs each

For no SMT, MPI Tasks = $\frac{128}{\text{Threads per MPI Task}}$

For SMT, MPI Tasks = $2*\frac{128}{\text{Threads per MPI Task}}$
Summary

• Monitoring Synchronisation
  – Might throw away 10% performance without it

• Binding
  – up to 20% performance enhancement

• SMT
  – typically 20% performance enhancement