CFD on the Intel Many Integrated Core Architecture

Christian Simmendinger
T-Systems Solutions for Research
Overview

CFD on Intel KNC
- TAU
- TRACE

Intel MIC Architecture
- Opportunities in CFD
- Challenges in CFD

Scalability
- *Core Level*
  SIMD and how to make use of it – Scout on Intel MIC.
- *Thread Level*
From X86 to Intel MIC
Opportunities in CFD

<table>
<thead>
<tr>
<th>X86 Intel Sandy Bridge</th>
<th>Intel MIC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Opportunities</strong></td>
<td></td>
</tr>
<tr>
<td>Peak: ~371 DP Gflop/sec</td>
<td>Peak: &gt; 1000 DP Gflop/sec</td>
</tr>
<tr>
<td>Bandwidth: ~79 GB/sec (Triad)</td>
<td>Bandwidth: 175GB/sec (Triad)</td>
</tr>
</tbody>
</table>

**Programmability, Portability, Performance Portability, Maintainability**

Opportunities in CFD

- TAU Code: >>100K lines of code, flat profile – no hotspots.
- TRACE Code: >>100K lines of code, flat profile – no hotspots.
From X86 to Intel MIC
Challenges in CFD

<table>
<thead>
<tr>
<th>X86 Intel Sandy Bridge</th>
<th>Intel MIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenges</td>
<td>8/16 Threads (HT)</td>
</tr>
<tr>
<td>256bit SIMD AVX</td>
<td>512bit SIMD</td>
</tr>
<tr>
<td>$8 \times 4 \times 2 = 64$ DP Flops/Cycle</td>
<td>$50 \times 8 \times 2 = 800$ DP Flops/Cycle</td>
</tr>
</tbody>
</table>

Challenges in CFD

- A massive increase in Thread Level Parallelism (> 200 Threads /Die)
- A moderate increase in Instruction Level Parallelism (16 DP Flops/Core)
Scalability Challenges
Core Level

**Programmability, Portability, Performance Portability, Maintainability**

- SSE2/SSE4.2/AVX/MIC: From 128 bit SIMD to 512 bit SIMD. 16 DP Flop/Cycle – or just one DP Flop/Cycle.
- Compiler performance doubles every 18 years. Do not expect compilers to resolve the SIMD issue for you. Especially C/C++ takes a severe performance hit due to aliasing.

Leverage SIMD intrinsics – but retain programmability, portability, performance portability and maintainability of the source code.

```c
#pragma scout loop vectorize
for (k = sk; k < ek; k++)
{
}
```
Scalability Challenges
Core Level

void g(float* a, float b, float* c)
{
    int i;
    #pragma scout loop vectorized
    for (i = 0; i < 100; ++i)
    {
        c[i] = a[i] + b;
    }
}

void g(float* a, float b, float* c)
{
    __m128 art_vectorized0, art_vectorized2,
    __art_vectorized1;
    int i;
    art_vectorized1 = _mm_set1_ps(b);
    for (i = 0; i < 100 - 3; i += 4)
    {
        art_vectorized0 = _mm_loadu_ps(&a[i]);
        art_vectorized2 =
            _mm_add_ps(art_vectorized0,
                        art_vectorized1);
        _mm_storeu_ps(&c[i]), art_vectorized2);
    }
    for (; i < 100; ++i)
    {
        c[i] = a[i] + b;
    }
}

warning: start processing
read_once.c:6:3: note: vectorizing efficiency: 3 vectorized ops, 0 unrolled ops
read_once.c:6:3: note: loop vectorized {tgt:14:18}
Scalability Challenges
Thread Level

Hyperplane Parallelisation / ILU

```c
for (hp = 0; hp < all_planes; hp++)
{
    #pragma omp for nowait
    for (cnt = start[hp]; cnt < stop[hp]; cnt++)
    {
        int mx0 = midx[cnt][0];
        int mx1 = midx[cnt][1];
        int px0 = pidx[cnt][0];
        int px1 = pidx[cnt][1];
        U[cnt] = (U[px0]+U[px1]+U[mx0]+U[mx1]) * 0.25;
    }
    #pragma omp barrier
}
```
Scalability Challenges
TRACE – Kernel Results

Amplifier/Vtune
• SIMD Execution - SCOUT
Scalability Challenges
TRACE – Kernel Results

Amplifier/Vtune
- SIMD Execution – SCOUT Gather/Scatter
Scalability Challenges
Core Level/Thread Level

Speedup TRACE

1 Core KNC ES1
Max Speedup KNC
Max Speedup KNC SCOUT

---

<table>
<thead>
<tr>
<th>CPU Time</th>
<th>Module</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.22.3151</td>
<td>libx Window</td>
<td>libx</td>
</tr>
<tr>
<td>7.15.0646</td>
<td>kmp_wait_sleep _mp5.so</td>
<td>libx</td>
</tr>
<tr>
<td>14.0.1440</td>
<td>kmp_int.so.4</td>
<td>libx</td>
</tr>
<tr>
<td>14.0.1440</td>
<td>TRACE</td>
<td>libx</td>
</tr>
</tbody>
</table>
Summary

CFD on Intel MIC Architecture
• Challenges and Opportunities

Scalability
• Core Level - SIMD – pretty good.
• Thread Level - not too bad.

Memory Subsystem
• Not yet there