Petascale in CFD
Cetraro, 29th of June, 2011

Christian Simmendinger, T-Systems Solutions for Research
Overview

The TAU Solver

Scalability
- Core (Instruction Level)
  - SIMD – SRC-SRC
- Thread (Task Level)
  - Beyond OpenMP
- System Level
  - GASPI / PGAS API

Maintainability of the Code
- Programmability
- Performance Portability
The TAU Solver
Vision: Digital Aircraft

TAU: 3D Finite Volume Reynolds Averaged Navier Stokes (RANS) Solver

Full flight envelope coverage: CFD mostly done near cruise point

configurations:
- clean
- airbrakes deployed
- high lift
- attached flow
- separated flow, unsteady

≈ 20,000,000 simulations

Engineering experience for current configurations and technologies

≈ 100,000 simulations
Programmability, Maintainability and Performance Portability: Instruction Level

SSE2/SSE4.2/AVX/MIC: From 128 bit SIMD to 512 bit SIMD.
16 DP Flop/Cycle – or just one.

• Statistical Fact:
  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.

SIMD Intrinsics

Scalability
• Core (Instruction Level)
  • SIMD – SRC-SRC
• Thread (Task Level)
  • Beyond OpenMP
• System Level
  • GASPI / PGAS API

• Maintainability
• Programmability
• Performance-Portability
Programmability, Maintainability and Performance Portability: Instruction Level
Programmability, Maintainability and Performance
Portability: Thread Level

- From MPI to OpenMP:
  - Resolve race conditions with thread pools and Mutual Exclusion of thread chunks.
  - Sub-domains in unstructured meshes

Portability:
- Thread Level
  - Core (Instruction Level)
  - SIMD – SRC – SRC
  - Beyond OpenMP
  - System Level
  - GASPI / PGAS API
Programmability, Maintainability and Performance Portability: Thread Level

- Mutual Exclusion

```c
for (color = fcolor; color != NULL; color = color->succ)
    for (face = color->start; face < color->stop; face++)
```

```c
for(color=get_color();color != NULL; color=get_next_color(color))
    for (face = color->start; face < color->stop; face++)
```

- Maintainability
- Programmability
- Performance Portability
Programmability, Maintainability and Performance Portability: Thread Level

OpenMP – The Fork Join Model

- Incremental Parallelisation
- Amdahl Trap (Barrier and Load Imbalance)
Beyond OpenMP:
• Resolve Amdahl’s trap with thread pools and a Mutual Completion of thread chunks.
• Replace global barriers with local data dependencies.

Programmability, Maintainability and Performance Portability: Thread Level
Programmability, Maintainability and Performance Portability: Thread Level

- **Mutual Completion**

```c
for (color = get_color(); color != NULL; color = get_next_color(color))
    for (face = color->start; face < color->stop; face++)
        BARRIER;
for (pnt = 0; pnt <nown; pnt++)
```

```c
for (color = get_color(); color != NULL; color = get_next_color(color))
    for (face = color->start; face < color->stop; face++)
        while (get_pnt_range(&pstart, &pstop))
            for (pnt = pstart; pnt < pstop; pnt++)...
```

- Maintainability
- Programmability
- Performance Portability
Programmability, Maintainability and Performance Portability: Thread Level

This Programming Model is **Generic**.

```c
for(k = 0; k < KDIM; k++)
    for (j = 0; j < JDIM; j++)
        for (i = 0; i < IDIM; i++) {}
BARRIER;
for(k = 0; k < KDIM; k++) ...
```

```c
for(k = get_k_index(); k != -1; k = get_next_k_index(k))
    for (j = 0; j < JDIM; j++)
        for (i = 0; i < IDIM; i++) {}
    for(k = get_k_index(); k != -1; k = get_next_k_index(k)) ...
```

Programmability, Maintainability and Performance Portability: Thread Level

This extension of the OpenMP Fork-Join Programming Model is

- fully asynchronous
- data flow driven
- implicitly load balanced
- exclusively uses local locks instead of global barriers
- expected to scale to $O(10^2)$ cores
Programmability, Maintainability and Performance Portability: System Level

In a Partitioned Global Address Space every thread can read/write the entire global memory of the application.

Opportunities for improved programmability
(PGAS languages like UPC, CAF, Chapel and others)

Opportunities for improved **Scalability**
(GPI (Fraunhofer ITWM), GASPI)

Global Address Space Programming Interface (GASPI)

A PGAS API which aims at replacing bulk-synchronous communication with asynchronous, 1-sided, RDMA driven communication patterns.

---

**Scalability**
- Core (Instruction Level)
  - SIMD – SRC-SRC
- Thread (Task Level)
  - Beyond OpenMP
- System Level
  - GASPI / PGAS API
Programmability, Maintainability and Performance Portability: System Level

Overlapping Communication and Computation

```c
if (this_thread_exchanges_pnt_data(g)) /* lock halo */
{
    exchange_pointdata();
    post_exch_pnt_data(); /* unlock halo*/
}
```
Folie 16

Programmability, Maintainability and Performance
Portability: System

Multithreaded send/recv

```c
if ((sid = get_seq_num_and_lock(g)) < num_send_threads(g))
    send_dbl_threaded(comap, comap->ncommdomains, comap->commpartner,
                       data, dim2, key);

If ((sid = thread_seq_num()) < num_recv_threads(g))
    recv_dbl_threaded(g, num_recv_threads, sid,
                      comap, cocomap->ncommdomain, cocomap->commpartner, data, dim2, key);
```

T1: writeDmaVM

T2: Transfer finishes
Programmability, Maintainability and Performance Portability: System Level

Strong Scaling Use Case: **How far can we scale?**

4W Multigrid, 1 Iteration Step.
- 1 Allreduce
- 140 Next Neighbour Halo Exchanges
Linear scaling for SG
8328 pnts/domain
1388 pnts/core
+4000 pnts halo

80 pnts/coarse lvl domain
13 pnts/core
+160 pnts halo

319 Gflops/sec
Linear scaling for SG
8328 pnts/domain
1388 pnts/core
+4000 pnts halo

80 pnts/coarse lvl domain
13 pnts/core
+160 pnts halo

319 Gflops/sec
Summary

Scalability

• Core (Instruction Level)
  • SIMD – SRC-SRC
• Thread (Task Level)
  • Beyond OpenMP
• System Level
  • GASPI / PGAS API / GASPI

We are confident, that we can reach $O(\text{Petascale})$ within the constraints of a typical production run.