European exascale applications workshop, Manchester, 11th and 12th October 2016

**DLR TAU-Code - Application in INTERWinE**

Thomas Gerhold, Barbara Brandfass, Jens Jägersküpper, DLR
Christian Simmendinger, T-Systems SfR
DLR TAU-Code - CDF Solver for External Aerodynamic

2nd Order Finite Volume method for unstructured grids & compressible flows

Used in European aircraft industry, research organizations and academia
Europe’s Vision for Aviation

- Maintaining Global Leadership & Serving Society’s Needs

Goals (relative to typical aircraft in 2000)

- CO$_2$ emissions reduced by 75%
- NOx emissions reduced by 90%
- 65% reduction in perceived aircraft noise

Consequence

- Heavy demands on future product performance
- Step changes in aircraft technology required
- New design principles mandatory
Numerical Simulation
Key Enabler for Future Aircraft Design

Future aircraft
- Design may be driven by unconventional layouts
- Flight characteristics may be dominated by non-linear effects

High-fidelity methods indispensable for design & assessment of step changing aircraft
- Reliable insight to new aircraft technologies
- Comprehensive sensitivity analysis with risk & uncertainty management
- Best overall aircraft performance through integrated aerodynamics / structures / systems design
- Consistent and harmonized aerodynamic and aero-elastic data across flight envelope

Further improvement of simulation capability necessary
HPC for future architectures is a key element
HPC for Aircraft Design

Challenge: consider algorithmic and parallel efficiency and trade off

Collaboration of experts in application domain and parallel computing necessary to extract – or even generate – data parallelism in algorithms

Target: \#iterations \times \text{wall clock / iteration}
HPC for Aircraft Design

Status

TAU: pure MPI Parallelization (typical daily grid sizes 20-60 Mpoints)

Scaling of TAU (RK, 31 mio points) on C^2A^2S^2E cluster Ivy Bridge 2x12 cores

- 80% parallel efficiency (vs 24 cores)
- Scalability limit ~6.5K points/core

Note: time to solution
- 4W multigrid vs. SG gain ~factor 5-10
HPC for Aircraft Design
Experiences: Hybrid Parallelization
1st TAU-prototype

Reduce MPI & domain decomposition overhead

Approach: Hybrid Shared-Memory / MPI Parallelization
- 1 MPI domain per chip (instead per core) increases sizes of MPI domains
- Reduces relative size of grid overlap & load imbalances
- Reduces number of messages and increases message sizes
- Asynchronous shared-memory parallelization to avoid the Amdahl trap

Concept: Task-Based Parallel Programming Model
- one single folk & join of thread-pool per Runge-Kutta cycle (1 OpenMP pragma)
- each (MPI-)domain is split into subdomains (colors) without overlap
- # subdomains >> # threads
- “task“: processing of a single color
- thread-pool model: dynamic allocation of “tasks“ to the threads
  - task dependencies (different tasks share grid points)
  - task dispatching allows for dynamic load-balancing
- Mutual exclusion of neighboring colors to prevent data races

Color Loop  (cache-blocked loop over the edges)

```c
for (color = grid->gedat->fcolor; color != NULL; color = color->succ)
```

```c
for (color = get_color(grid); color != NULL; color = get_next_color(color, grid))
```

Task dispatching & mutual exclusion (thread management)
Point Loop (here: over the ownpoints, i.e. w/o ghost points)

```c
for(pnt = 0; pnt < grid->gpdat->nownpoints; pnt++)

while(get_pnt_range_nown(grid, &pnt, &i))
    for(; pnt < i; pnt++)
```
Synchronize the halos (i.e. the ghost points) with MPI

```
exchange_pointdata(grid, pgrad, 3 * N_PRIM_GRAD, GRADIENTS);

if(this_thread_exchanges_pnt_data(grid))
{
    exchange_pointdata(grid, pgrad, 3 * N_PRIM_GRAD, GRADIENTS);
    post_exch_pnt_data(grid);
}
```

Overlap communication with computation

Halo-touching colors excluded from processing while halo data is being updated using MPI
Performance hybrid-parallel TAU – Improved Scaling

Grid: 13 Mio Points (Highlift Configuration)
3-stage Runge/Kutta, JST, scalar dissipation, GG gradients, SA turbulence

Cluster: C²A²S²E² II
- Intel Westmere EP
- 6-core CPU @ 3GHz
- HyperThreading OFF
- 2 CPUs per node
- QDR Infiniband (M9)

REF: #Domains = #Cores = 12 * #Nodes
GASPI vs. MPI

- Replace MPI by GPI (GPI is a PGAS-API from Fraunhofer ITWM)
  Initially: C²A²S²E Cooperation with T-System SfR & Fraunhofer ITWM
  Continued: www.GASPI.de, a national funded Project by BMBF

GPI allows for real one-sided communication
- no hand shake as in MPI protocol
- enables for asynchronous communication → no global sync
- no CPU clock cycles needed for send (send is an IB hardware call)
- enables for complete overlap of communication & computation

Code modifications to profit from one-sided communication
- reorder faces and points, such that halo data is computed first
- split send and receive: start send after halo data is computed
- compute inner data next in overlap with communication
- receive: check data packages in “post office”
  Ideally, data already available else wait (local sync only)
PGAS vs. MPI

- Replace MPI by GPI (GPI is a PGAS-API from Fraunhofer ITWM)
- Implementation of Prototype by C. Simmendeder, T-Systems SfR

### Benchmarks on C²A²S²E II

<table>
<thead>
<tr>
<th>Points/core</th>
<th>F6, 4W Multigrid, 2 Mill. Points</th>
<th>F6, SG, 2 Mill. Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td><img src="TAU-ShMem/GPI" alt="Graph" /> ~50%</td>
<td><img src="TAU-ShMem/GPI" alt="Graph" /> ~100%</td>
</tr>
<tr>
<td>2500</td>
<td><img src="TAU-ShMem/MPI" alt="Graph" /></td>
<td><img src="TAU-ShMem/MPI" alt="Graph" /></td>
</tr>
<tr>
<td>1400</td>
<td><img src="TAU-MPI" alt="Graph" /></td>
<td><img src="TAU-MPI" alt="Graph" /></td>
</tr>
</tbody>
</table>

(fine grid)
Hybrid parallelized TAU vs. TAU-MPI on Intel MIC

SharedMem Parallelization for HyperThreading per core compared to 1MPI-domain per HyperThread

Note: SharedMem Parallelisation over a set of cores is less efficient

Frequent access to shared data (via ring-bus) for atomic locks of colors is too expensive.

NOTE: NO SIMD vectorization!
F6 Full Model 31 mio mesh points, 3 V Mesh:

- 1345 mesh points / thread @level 1
- 180 mesh points / thread @level 2
- 24 mesh points / thread @level 3

Strong Scaling

- linear
- comm_free
- mpi_bsync_se_gs
- mpi_early_recv_se_gs
- mpi_async_se_gs
- gaspi_bsync_gs
- gaspi_async_gs
- mpi_fence_bsync_se_gs
- mpi_fence_async_se_gs
- mpi_pscw_bsync_se_gs
- mpi_pscw_async_se_gs
TAU CFD-Proxy Xeon Phi
EXA2CT Christian Simmendinger

F6 2 mio mesh points, 3 V Multigrid
- 260 mesh points / thread @level 1
- 35 mesh points / thread @level 2
- 5 mesh points / thread @level 3

Strong Scaling Near Exascale

- linear
- comm_free
- mpi_bsync_se_gs
- mpi_early_recv_se_gs
- mpi_async_se_gs
- gaspi_bsync_gs
- gaspi_async_gs
- mpi_fence_bsync_se_gs
- mpi_fence_async_se_gs
- mpi_pscw_bsync_se_gs
- mpi_pscw_async_se_gs

0 2000 4000 6000 8000 10000
0 5 10 15 20 25 30 35
Multi-threaded pack of data to be aggregately sent (written remotely) is crucial to scalability, in particular on Xeon Phi
DLR Next Generation Solver *Flucs*

**Goal:** Secure Quality of CFD for Virtual Aircraft Requirements

- Full exploitation of future HPC systems
- Consolidation of current DLR CFD solvers
- Flexible building blocks
- Basis for innovative concepts & algorithms e.g. high-order-finite element discretization, adaptivity, ...
- Seamless integration into multidisciplinary simulation environment
- State-of-the-art software engineering methods (C++11, templates)
Multi-Level Parallelization: MPI/GPI + Threads

Classical Domain Decomposition (1st level) via GPI or MPI

- One ghost-points communication buffer per domain and neighbor domain
- Ghost-point data grouped by donor domain → no scattering for receives
- But: **Overlap communication** with computation on domain-local data
- **Multi-threaded pack of data** for halo exchange

Sub-Decomposition with overlap (2nd level) via Threads

- **Static tasks:** one-to-one mapping of a sub-domain to a (logical) core (thread)
  - Each element is and stays located r/w in one core‘s cache hierarchy
  - **Data locality**
- **Cross-subdomain faces**
  - Data left and right of the face are read by both subdomains’ threads
  - Writes are only on the owned elements
  - **Computations for such a face is done twice**
  - Minimum number of arithmetic operations traded for data parallelism
Flucs Parallel Efficiency – Strong Scaling Scenario
1.9 Mio Elements @ C²A²S²E Cluster

TAU @ 2 mio. points (noncomparative info)

#nodes used (each with two 12-core IVB EP)
Summary of experiences

TAU - MPI/GPI + OpenMP

asynchronous dynamic task model (some threads can be several loop ahead others)
mutual exclusion by atomic operations „too“ expensive on KNC
non-deterministic data flow is very hard to debug

Flucs MPI-GPI mixed + OpenMP

Flucs - MPI or GPI runs as a FlowSimulator- MPI Plug-in (in an epoche)
static tasks per subdomain (so far) for flux computation & residual update

GASPI scales beyond MPI (for our applications)
Work planned/ongoing in INTERTWinE

Experimenting

- with dynamic task models by using TAU kernels
  - to improve load balance between threads
  - load inbalances due to hierachical memory accesses
- TAU Proxy for computation of residuals: T-System SfR
- TAU Kernel for sparse linear algebra solvers: DLR
  (which we will implement in Flucs for using implicit methods)

- Interoperability issues between the different programming models: MPI/GASPI, OpenMP, OmpSs

Goal

- Use of results in development of TAU-successor Flucs