Hybrid parallelization of a pseudo-spectral DNS code and its computational performance on RZG’s iDataPlex system ”Hydra”

Markus Rampp\textsuperscript{1}, Liang Shi\textsuperscript{2}, Marc Avila\textsuperscript{3,2}, Björn Hof\textsuperscript{2,4}

\textsuperscript{1}Computing Center of the Max-Planck-Society and the IPP (RZG), Garching
\textsuperscript{2}Max-Planck-Institute for Dynamics and Self-Organization, Göttingen
\textsuperscript{3}Friedrich-Alexander-Universität Erlangen-Nürnberg
\textsuperscript{4}Institute of Science and Technology (IST) Austria

ScicomP 19, Lugano, May 2013
Introduction
  direct numerical simulations of the Navier-Stokes equation
  Taylor-Couette flow and turbulence in accretion discs

The code \texttt{nsCouette}
  overview
  hybrid parallelization

Results
  performance on IBM iDataPlex
  some technical remarks

Conclusions and outlook
  PRACE/DECI project HYDRAD
Navier Stokes equation

Navier Stokes equation for incompressible fluid \((\nabla \cdot \mathbf{u} = 0)\)

\[
\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \Delta \mathbf{u}, \quad \nabla \cdot \mathbf{u} = 0
\]

\(\mathbf{u}\): velocity, \(p\): pressure, \(\rho\): density, \(\nu := \frac{\eta}{\rho}\): kinematic viscosity

Reynolds number (fundamental dimensionless parameter characterizing the flow):

- \(Re := \frac{UL}{\nu}\) \(U, L\): characteristic velocity, length scales
- \(Re \approx \left| \mathbf{u} \cdot \nabla \mathbf{u} \right| / |\nu \Delta \mathbf{u}| \approx \text{inertial forces/viscous forces}\)
- relevance:
  - similarity scaling with \(Re\) (e.g. wind tunnels, . . .)
  - \(Re\) determines transition from laminar to turbulent flow

Direct numerical simulations of turbulent flow

On the nature of turbulence (”the turbulent cascade”): 

Big whorls have little whorls
That feed on their velocity,
And little whorls have lesser whorls
And so on to viscosity.

L. F. Richardson, ∼ 1920,
alluding to a poem by J. Swift (1667-1745)

Direct numerical simulations (DNS):

- solve discretized Navier-Stokes equation without assuming a turbulence model (cf. LES, RANS, . . .)
- all relevant spatial and temporal scales must be resolved: e.g. from $L$ down to the energy dissipation (Kolmogorov) scale $(\nu^3/\dot{\epsilon})^{1/4} \Rightarrow \cdots \Rightarrow N_{\text{grid points}} \sim Re^{9/4}, N_{\text{time steps}} \sim Re^{3/4}$

$\sim$ total numerical complexity of a DNS in 3 dimensions: $\mathcal{O}(Re^3)$

$\sim$ computationally demanding for high $Re$
Application: Taylor-Couette flow

Taylor-Couette (TC) flow: a classical hydrodynamics problem

- Viscid fluid between two concentric, rotating cylinders
- Development of "Taylor vortices": $Re_{\text{inner}}, Re_{\text{outer}}$

- Analytic and numerical modelling
- Experiments: fluid with marker particles, end walls (!)
- Measure torque, observe flow patterns, ...
Specific motivation of this project (PIs: B. Hof, M. Avila)

- relevance of Taylor-Couette flow:
  - fundamental research on turbulence (TC flow, pipe flow, ...)
    - relevance: scientific, technical, economical, ...
      (e.g. Avila PRL 2012, Hof et al. Science 2011, ...)
  - a model for astrophysical accretion disks
    - relevance: formation of stars and planets
      revived by conflicting results from laboratory experiments

- numerical modelling requires DNS of the Navier-Stokes equation up to $Re \simeq 10^5 \ldots 10^6$
  ⇒ an HPC problem (recall: $O(Re^3)$)

we need:
  - efficient and scalable code
  - sufficient computing resources
A long-standing controversy about whether the motions within a typical astrophysical disk of gas are stable or unstable has resurfaced. The answer has profound significance for our understanding of how stars and planets form.

[...]
Understanding how a fluid with internal motion makes a transition from smooth (laminar) flow to turbulence remains a stubborn, long-standing theoretical challenge to the hydrodynamics community. A particularly relevant astrophysical application of this problem is the study of star formation, because stars similar to the Sun are thought to pass the earliest stages of their lives forming at the centres of gaseous 'protostellar' disks. The question is whether the gas, which orbits in nearly circular motion (much like the planets orbiting the Sun), is a turbulent or laminar fluid. [...]
Because of its central importance to astrophysics, the possibility that disks may be turbulent for purely hydrodynamical reasons will probably excite another round of intense investigative activity, both in the laboratory and on the computer.[...]
Protoplanetary accretion disks

- canonical standard models "$\alpha$-Disc" (Shakura & Sunyaev, 1973), "MRI" (Balbus & Hawley, 1991)
- how can angular-momentum transport be explained physically?
- hydrodynamic turbulence observed in Taylor-Couette experiments *could* explain rates $dL/dt, dM/dt$ (Paoletti et al. 2011, 2012)
- conflicting experimental results: artifacts due to finite-size effects?

\[ \rightarrow \]\ investi\-gate hydrodynamic stability of cool Keplerian discs with DNS
\[ \rightarrow \]\ preliminary tests suggest strong non-linear growth of finite-amplitude perturbations
nsCouette: high-level overview

The code \textit{nsCouette}:

- a classical pseudo-spectral discretization
- $2^{\text{nd}}$-order, semi-implicit timestepping algorithm (Hugues & Randriamampianina, 1998)
- tailored to simulations of Taylor-Couette flow
- fixed grid in 3 dimensions, cylindrical coordinates $(r, \theta, z)$
- originally developed and implemented by M. Avila & L. Shi (plain MPI)
- joint development and optimization of hybrid MPI/OpenMP code (Shi, Rampp, Hof, Avila \textit{in preparation})
Spatial discretization scheme ("pseudo-spectral"): 

- \( \partial_t u + u \cdot \nabla u = -\frac{1}{\rho} \nabla p + \nu \Delta u, \quad \nabla \cdot u = 0 \) (Navier-Stokes equation)
- cylindrical coordinates \((r, \theta, z)\)
- use FD in \(r\), spectral scheme for \((\theta, z)\) (periodic BC)
- expansion of \((u, p)\) in Fourier series, e.g.
  \[
  p(r, \theta, z) = \sum_{l=-L}^{L} \sum_{n=-N}^{N} \hat{p}^{ln}(r) e^{i(lk_z z + nk_\theta \theta)}
  \]
  \(\leadsto\) 3 Helmholtz equations for \(\hat{u}\), decoupled in Fourier space
  \(\leadsto\) 2 Poisson equations for \(p\) \((\Leftarrow \nabla \cdot u = 0)\)
- nonlinear term \(u \cdot \nabla u\) is computed in ’real space’ to avoid \(O(N^2)\) convolution in Fourier space
Spatial discretization scheme ("pseudo-spectral"):  
- $\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \Delta \mathbf{u}, \ \nabla \cdot \mathbf{u} = 0$ (Navier-Stokes equation)  
- cylindrical coordinates $(r, \theta, z)$  
- use FD in $r$, spectral scheme for $(\theta, z)$ (periodic BC)  
- expansion of $(\mathbf{u}, p)$ in Fourier series, e.g.  
  $$p(r, \theta, z) = \sum_{l=-L}^{L} \sum_{n=-N}^{N} \hat{p}^l n (r) e^{i(lk_z z + nk_\theta \theta)}$$  
  $\Rightarrow$ 3 Helmholtz equations for $\hat{\mathbf{u}}$, decoupled in Fourier space  
  $\Rightarrow$ 2 Poisson equations for $p$ ($\Leftarrow \nabla \cdot \mathbf{u} = 0$)  
- nonlinear term $\mathbf{u} \cdot \nabla \mathbf{u}$ is computed in 'real space' to avoid $O(N^2)$ convolution in Fourier space

Temporal scheme: for all Fourier modes $(l, n)$ do . . .  
1. compute nonlinear term as $\text{FFT}^{-1}\{\text{FFT}\{\hat{\mathbf{u}}\} \cdot \text{FFT}\{\nabla \hat{\mathbf{u}}\}\}$  
2. pressure prediction: solve Poisson equation $\Delta \hat{p} = \ldots$  
3. velocity prediction: solve Helmholtz equations $\Delta \hat{\mathbf{u}} + k^2 \hat{\mathbf{u}} = \ldots$  
4. corrector step (another Poisson equation)
nsCouette: parallelization scheme

Principle: exploit parallelism in Fourier modes \((l, n)\)

- distribute Fourier modes \((\hat{u}_{l,n})\) to processors (MPI-tasks)
- solve linear equations independently for each Fourier mode \((l, n)\)
- perform global transposition \((k, l, n) \mapsto (l, n, k)\)
- compute FFTs and computation of nonlinear term independently for each radial point \(k\)

- straightforwardly implemented with plain, 1D ("slab"), MPI domain decomposition of \((K, L \cdot N)\)-grid
nsCouette: hybrid parallelization

- typically \( K \ll L \cdot N \):
  - \( K \lesssim 500 \) radial points
  - \( L \cdot N \approx 10^5 \ldots 10^6 \) Fourier modes

⇒ number of MPI-tasks limited by number of radial points
⇒ add OpenMP parallelization (or implement a more sophisticated MPI scheme)
nsCouette: hybrid parallelization

- typically $K \ll L \cdot N$:
  - $K \lesssim 500$ radial points
  - $L \cdot N \approx 10^5 \ldots 10^6$ Fourier modes

⇒ number of MPI-tasks limited by number of radial points
⇒ add OpenMP parallelization (or implement a more sophisticated MPI scheme)

use OpenMP threads for parallelism in $(l, n)$
loop-level: $\approx 500$ iterations per MPI-task

use OpenMP threads for some parallelism in $(k)$
task-level: $\approx 2\ldots 8$ OpenMP-tasks (FFTs and gradients of individual components) per MPI-task
Limits

- speedup $\leq n_{\text{cores}} \leq K \simeq 500$ (original MPI code)
- speedup $\leq n_{\text{cores}} \leq K \cdot n_{\text{cores per node}} \simeq 500 \cdot 16$ (hybrid code)
Limits

- speedup $\leq n_{\text{cores}} \leq K \simeq 500$ (original MPI code)
- speedup $\leq n_{\text{cores}} \leq K \cdot n_{\text{cores per node}} \simeq 500 \cdot 16$ (hybrid code)

Implementation

- language standards: Fortran 90, MPI, OpenMP
- platforms: x86_64 HPC clusters, IBM Power6, IBM BlueGene/P
- status: in production

Performance-critical ingredients, libraries

- linear solvers (dimension $\simeq 500$, bandwidth $\simeq 4$): GBTRF, GBTRS from Intel MKL
- 2-dimensional FFTs (r2c, c2r): from FFTW (or MKL)
- global transpose: MPI_Alltoall and task-local transpose()
- parallel I/O using pHDF5 (collective mode)
Benchmark setup

Hardware: HPC system of the MPG ’Hydra’

- 610 nodes w/ 2 Xeon E5-2670 ”Sandy Bridge”, operated @ 2.6 GHz
- 2 · 8 (physical) cores per node
- FDR 14 InfiniBand interconnect
- first phase of a (much larger) IBM deployment at RZG/MPG, 2013

Software

- Intel compilers (13.1) and MKL (11.0)
- IBM PE (1.2)
- FFTW (3.3.3), pHDF5 (1.8.9)

Benchmarks

- strong scaling study of 2 relevant setups: SMALL, LARGE
- focus: OpenMP efficiency (MPI scaling already very good)
Strong scaling overview

**SMALL setup**
- $n_r = 32 = \max \{\# \text{ MPI-tasks}\}$
- $n_f = n_\theta \cdot n_z = 384 \cdot 640$
- relevance/application: localized turbulence studies ($Re \approx 10^2$)

**LARGE setup**
- $n_r = 512 = \max \{\# \text{ MPI-tasks}\}$
- $n_f = n_\theta \cdot n_z = 256 \cdot 1024$
- relevance/application: TC flow in Keplerian discs ($Re \approx 10^6$)
Parallel efficiency of OpenMP parallelization

\[ \eta := \frac{T_n}{n \cdot T_1}, \quad n: \text{ number of OpenMP threads per task (1...16)} \]

### SMALL setup (32, 384, 640)

<table>
<thead>
<tr>
<th>tot. cores (n)</th>
<th>32(1)</th>
<th>64(2)</th>
<th>128(4)</th>
<th>256(8)</th>
<th>512(16)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(T_1) [s] (\eta)</td>
<td>(T_2) [s] (\eta)</td>
<td>(T_4) [s] (\eta)</td>
<td>(T_8) [s] (\eta)</td>
<td>(T_{16}) [s] (\eta)</td>
</tr>
<tr>
<td>nonlinear (1)</td>
<td>0.760 100%</td>
<td>0.435 87%</td>
<td>0.265 72%</td>
<td>0.161 59%</td>
<td>0.105 45%</td>
</tr>
<tr>
<td>(p) prediction (2)</td>
<td>0.084 100%</td>
<td>0.042 99%</td>
<td>0.021 100%</td>
<td>0.011 97%</td>
<td>0.006 84%</td>
</tr>
<tr>
<td>(u) prediction (3)</td>
<td>0.218 100%</td>
<td>0.109 99%</td>
<td>0.055 98%</td>
<td>0.028 96%</td>
<td>0.016 83%</td>
</tr>
<tr>
<td>correction (4)</td>
<td>0.089 100%</td>
<td>0.044 101%</td>
<td>0.022 101%</td>
<td>0.011 99%</td>
<td>0.006 90%</td>
</tr>
<tr>
<td>complete step</td>
<td>1.217 100%</td>
<td>0.666 91%</td>
<td>0.385 79%</td>
<td>0.229 66%</td>
<td>0.152 50%</td>
</tr>
</tbody>
</table>

### LARGE setup (512, 256, 1024)

<table>
<thead>
<tr>
<th>tot. cores (n)</th>
<th>512(1)</th>
<th>1024(2)</th>
<th>2048(4)</th>
<th>4096(8)</th>
<th>8192(16)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(T_1) [s] (\eta)</td>
<td>(T_2) [s] (\eta)</td>
<td>(T_4) [s] (\eta)</td>
<td>(T_8) [s] (\eta)</td>
<td>(T_{16}) [s] (\eta)</td>
</tr>
<tr>
<td>nonlinear (1)</td>
<td>1.20 100%</td>
<td>0.67 90%</td>
<td>0.33 91%</td>
<td>0.20 75%</td>
<td>0.19 39%</td>
</tr>
<tr>
<td>(p) prediction (2)</td>
<td>1.12 100%</td>
<td>0.49 114%</td>
<td>0.25 112%</td>
<td>0.13 108%</td>
<td>0.08 88%</td>
</tr>
<tr>
<td>(u) prediction (3)</td>
<td>1.38 100%</td>
<td>0.61 113%</td>
<td>0.31 111%</td>
<td>0.17 101%</td>
<td>0.11 78%</td>
</tr>
<tr>
<td>correction (4)</td>
<td>1.12 100%</td>
<td>0.52 108%</td>
<td>0.28 100%</td>
<td>0.15 93%</td>
<td>0.09 78%</td>
</tr>
<tr>
<td>complete step</td>
<td>4.88 100%</td>
<td>2.32 106%</td>
<td>1.20 102%</td>
<td>0.66 92%</td>
<td>0.49 62%</td>
</tr>
</tbody>
</table>
Global transposition routine (acknowledgement: Florian Merz, IBM)

- Global transposition $(k, l, n) \mapsto (l, n, k)$ in processor space
- Implemented with MPI_Alltoall + local transpose
- Can be (partially) overlapped with computations in nsCouette using threads
- Issues: MPI_Alltoall
  - Ultimately limits scaling of the code
  - MPI_Alltoall from PE (1.2) is slower by 2x wrt. PEMPI, IMPI
  - Relevant message size LARGE setup: $\approx 8$ kB
  - ... acknowledged by IBM
- Issues: task-local out-of-place transpose
  - Performed in serial mode
  - mkl_zomatcopy from Intel MKL (11.0.3) can be significantly slower than Fortran intrinsic transpose() (Intel ifort 13.1.1)
  - ... acknowledged by Intel
Performance of MKL transpose

- `mkl_zomatcopy` (MKL 11.0.3) is outperformed by Fortran intrinsic `transpose()` (ifort 13.1.1)
- under investigation by Intel

**double complex:** `mkl_zomatcopy`

**double:** `mkl_domatcopy`
SIMD vectorization

Auto-vectorization of double complex data type

SUBROUTINE predictor_v()

    DO j=2,m_r-1
        aDiag(j) = A(j,j)
        A(j,j) = aDiag(j) + (1+2*f_hat_mp(j,k)%k_th)/r(j)**2
        !b(j) = - DOT_PRODUCT(Mw_dr(j,:),p_hat_mp(:,k)) &
        b(j) = - dotp(j) &
            + f_hat_mp(j,k)%k_th*p_hat_mp(j,k)/r(j) &
        - udu_plus(j,k) &
            + (4*u_plus(j,k)-u_plus_old(j,k))/(2*dt)
    END DO

SSE (128 bit, Nehalem) cannot vectorize double-complex ops:

mod_timeStep.f90(211): (col. 8) remark: loop was not vectorized: data type unsupported on given target architecture.

AVX (256 bit, Sandy Bridge) can vectorize double-complex ops:

mod_timeStep.f90(211): (col. 8) remark: PARTIAL LOOP WAS VECTORIZED.
LAPACK vs. ESSL

SUBROUTINE luSolver(n,k,A,b,x)

#ifdef USE_ESSL
   call dgbf( AB,ldab,n,k,k,ipiv)
   call dgbss( AB,ldab,n,k,k,ipiv,par(:,1))
#else
   CALL dgbtrf(n,n,k,k,AB,ldab,ipiv,info)
   CALL dgbtrs('N',n,k,k,2,AB,ldab,ipiv,par,ldb,info)
#endif

...this is how this project started (RZG’s Power6, BlueGene/P) ...

Outer SIMD vectorization ?

• solve many independent systems with different RHS
• our simple, hand-written tridiagonal solver can outperform MKL
• an option (or rather a must ?) for MIC
Node performance

Roofline model for basic assessment

- visualize memory-bound vs. compute-bound
- "rooflines":
  hardware characteristics
- AI=Flops/Byte:
  algorithm characteristics

How to measure MB/s MFlop/s?

- tools: LIKWID, IBM’s, Intel’s, ...
- but: performance counters are broken in SandyBridge

(see Jan Treibig’s blog at http://likwid-tools.blogspot.de/2012/02/intel-sandybridge-and-counting-flops.html)
Summary

- developed a hybrid MPI/OpenMP code for DNS of Taylor-Couette flow
- elegant and natural extension of a plain MPI slab decomposition
- suited for multicore and manycore
- parallel scalability and efficiency enables highly resolved simulations at $Re \approx 10^5 \ldots 10^6$

Hydrodynamic stability of rotating flows in accretion discs (HYDRAD)

- 5 runs @ $Re \approx 10^5$
- resolution: $(512, 512, 1024)$
- timesteps/run: $2 \cdot 10^6$
- runtime per timestep $\approx 0.5 \text{ s} @ 8000$ cores
- total resources: 4 Million core hours (PRACE, MPG)
- simulations just started
- answered call for additional PRACE Tier-0 resources