CFD in water turbines – an HPC challenge

Håkan Nilsson

Chalmers, Dept. of Applied Mechanics, Div. of Fluid Dynamics, 412 96 Gothenburg
hani@chalmers.se,
WWW home page: http://www.tfd.chalmers.se/~hani

Abstract. 50% of the electric power in Sweden is generated by water power. Many of the power plants in Sweden are getting old and some major refurbishments are coming up. Due to the development of numerical methods and computer power the last decades Computational Fluid Dynamics (CFD) is to a large extent used as a design tool for this purpose. The general features of the flow in water turbines can be resolved with todays methods and computational power, but in order to study the flow in detail enormous HPC facilities and new methods are required.
The present work presents the water turbine field with its HPC requirements, shows some state-of-the-art results from OpenFOAM CFD analysis, and presents a parallel performance analysis on a Linux cluster using an ordinary gigabit interconnect v.s. an Infiniband interconnect.

1 Water turbines

Water turbines are designed to extract energy from the water flowing through the water turbine runner. The availabe power is determined by the difference in the elevation of the tail water and head water multiplied with the water mass flow and gravity. In reaction turbines the flow enters the runner with a swirl and the runner is designed to remove that swirl before the water enters the draft tube (see figure 1). The draft tube is a diffuser which recovers the static pressure and leads the water towards the tail water.

1.1 HPC challenges

Water turbines have complicated geometries where the flow in different parts of the turbine influences the flow in other parts of the turbine. To be able to set valid boundary conditions and to model the flow correctly it would thus be ideal to include all the geometry from head water to tail water. The Reynolds numbers in water turbine applications are always high, so the resolution of the computational mesh must be fine where large gradients in the flow occur. These requirements together yields an enourmously large mesh. In addition to this there are both stationary and rotating parts in water turbines, which requires rotor-stator interaction and unsteady computations. The thin wakes after stayvanes, guide vanes and runner blades affect the details of the flow and should also be resolved. Close to the runner blades cavitation often occurs, and the numerical methods to resolve those effects require even more cells and shorter time steps. There is probably no limit on the HPC requirements for flow in water turbines.

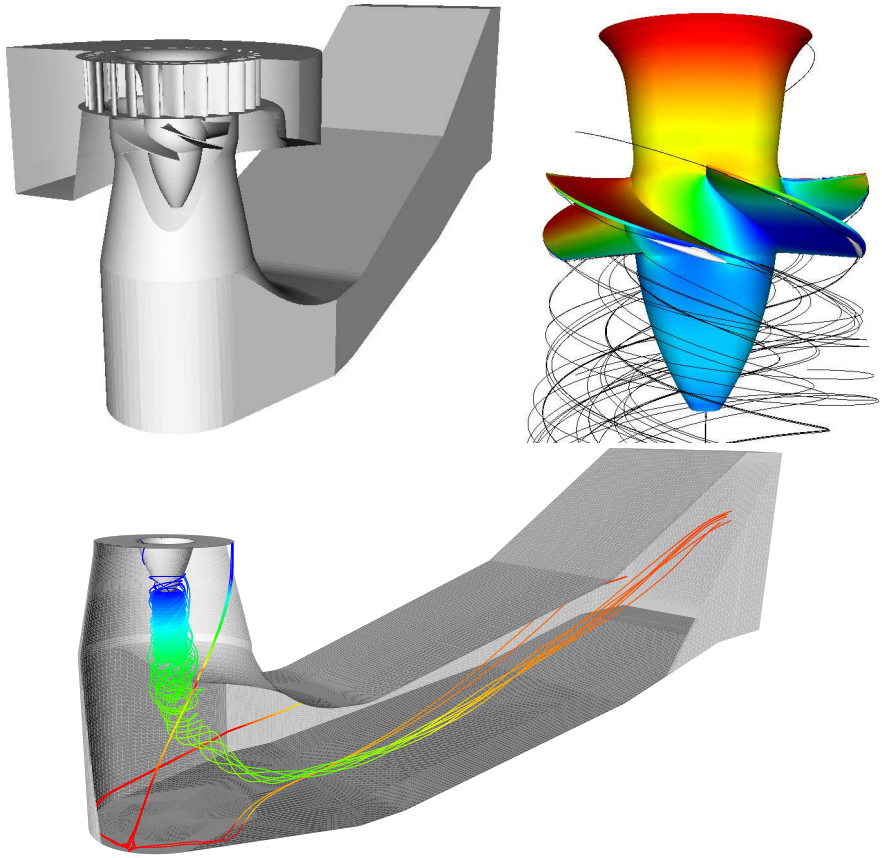

Fig. 1. CAD model of the Hölleforsen Kaplan turbine model and visualizations of the flow in the runner and draft tube.

2 The OpenFOAM CFD tool

The newly released OpenFOAM (Open Field Operation and Manipulation) tool has been used in the present study. OpenFOAM is an OpenSource object oriented C++ tool for solving arbitrary PDE's. It includes preprocessing (grid generator, converters, manipulators, case setup), postprocessing (using OpenSource Paraview) and many specialized CFD solvers are implemented. Some of the more specialized features that are included in OpenFOAM, and are important to the flow in water turbines are: sliding grid, moving meshes, two-phase flow (Langrange, VOF, Euler-Euler) and fluid-structure interaction. OpenFOAM runs in parallel using automatic/manual domain decomposition. In addition to the source code, OpenFOAM gives access to an international community of OpenFOAM researchers through the discussion board at the www.openfoam.org home page.

3 The studied cases

In the present work two parts of the Hölleforsen turbine model has been modeled, the runner and the draft tube (see figure 1). The two parts have been modeled separately, but the aim is to simultaneously model all parts including the spiral casing, the runner and the draft tube. The present runner computations have an inlet boundary condition from a previous computation of the flow through the guide vanes. Steady computations have been made for both a periodic part of the runner (1/5) and the whole runner. The present draft tube computations have inlet boundary conditions from detailed LDA measurements. Both steady [1] and unsteady computations have been made for the draft tube. In all cases the standard $k - \varepsilon$ turbulence model is used. The sizes of the meshes used are, 450 000 cells for the periodic runner computation, 2 240 000 cells for the full runner computation, and 1 000 000 cells for the draft tube computations. These are all wall-function grids that do not resolve the boundary layers in detail.

4 Parallel performance

A parallel performance test has been made using the draft tube case described above, with about 10^6 cells. The decompositions of the domain into 2, 4, 8 and 16 subdomains were made using the automatic load-balanced decomposition (Metis) in OpenFOAM. The Linux cluster was a 4 node Dual socket AMD Opteron 280 (2.4 GHz, dual core) with 4GB DDR400 RAM, i.e. 4 cores (CPUs) per node and a total of 16 cores (CPUs). Two different interconnects were tested, a Gigabit Ethernet through an HP ProCurve 2824 Switch, and an Infiniband (PCI-X) through a Silverstorm 9024 Switch. The SuSE Linux Enterprise Server, Service pack 3 operating system was used. The analysis has been made together with Peter Rundberg at Gridcore (www.gridcore.se).

The wall clock times were measured for three iterations (no I/O), and are presented in table 1. The table also shows the speed-up normalized by the single CPU run for each configuration, and the speed-up when using the Infiniband interconnect instead of the Gigabit interconnect (based on the speed-up columns in the tables). The reason for the difference between the single CPU runs is unknown, but its presence is not unexpected. Many things can influence computational speeds to this order of magnitude ($\sim$1%). The reason for the difference between the runs with 2 CPUs on 1 node is also unknown. The interconnect should not be important in this case. It has however been observed that sometimes when running two processes on the same node they end up at the same socket, which influences the computational speed to this order of magnitude ($\sim$10%). Later versions of the Linux kernel should have fixed this problem. An effect related to this can be seen in table 1, where the speed-up is increased if the computations are spread over as many nodes as possible. It is unknown why IBA has a worse tendency than ETH when distributing the 4 CPU case on different numbers of nodes. The more nodes involved, the more important the interconnect. The table however suggests that the Gigabit interconnect is the best in the 4 CPU case.

Table 1. Parallel performance using 1Gbit Ethernet (ETH) and Infiniband (IBA) interconnects. Packed vs. spread CPU distribution (the distribution of the processes on the nodes).

# CPU	# nodes	ETH (s)	IBA (s)	ETH (speed-up)	IBA (speed-up)	$\frac{IBA\ (speed-up)}{ETH\ (speed-up)}$ (based on speed-up)
1	1	165	163	1.0	1.0	1.0
2	1	86	78	1.9	2.1	1.1
2	2	85	81	1.9	2.0	1.0
4	1	76	72	2.2	2.3	1.0
4	2	64	62	2.6	2.6	1.0
4	4	53	56	3.1	2.9	0.9
8	2	43	41	3.8	4.0	1.0
8	4	41	35	4.0	4.7	1.2
16	4	23	20	7.2	8.2	1.1

5 Conclusions

The OpenFOAM CFD tool has proven to be a good platform for high quality computations of flow in water turbines. One major benefit of using OpenFOAM instead of any commercial CFD tool is that the full source code is available. This makes possible development of methods that are out of reach in the commercial tools. Another benefit is that OpenFOAM is free of charge, which makes possible international collaboration on a common platform without expensive licenses. The total expertice in the OpenFOAM discussion group is enormous. OpenFOAM is also a competitor with the commercial softwares in terms of advanced features. The included features grow rapidly through contributions from the advanced users, which makes OpenFOAM a true competitor to both commercial tools and in-house research codes.

Infiniband does not significantly improve the speed-up for this application, compared with the gigabit Ethernet interconnect. There is no obvious trend showing that Infiniband is preferable for all decompositions and process distributions. The largest improvement is seen when the case is distributed on as many CPUs as possible. When distributing the case on 16 CPUs the Infiniband interconnnect has a $\sim$10% speed-up compared with the corresponding gigabit Ethernet computation. Infiniband would probably benefit most from a distribution on more CPUs. The best configuration seems to be single CPU nodes connected with a high speed interconnect. However, this will affect the price of the system, the cooling requirements, the space requirements etc.

References

1. Nilsson, H., Page, M.: OpenFOAM SIMULATION OF THE FLOW IN THE HÖLLEFORSEN DRAFT TUBE MODEL. In Proceedings of Turbine-99 III (2005)