PROCEEDINGS OF SCIENCE

Status of the AuroraScience Project

Francesco Di Renzo[*†]
University of Parma and INFN
E-mail: francesco.direnzo@unipr.it

AuroraScience is a research project aiming at developing a computer architecture which benefits from both state of the art components/solutions (multi-core processors, liquid cooling, InfiniBand) and a custom network (an FPGA-based implementation of a 3-D torus). We report on the status of the project.

XXIX International Symposium on Lattice Field Theory
July 10 V 16 2011
Squaw Valley, Lake Tahoe, California

[*]Speaker.
[†]On behalf of the AuroraScience Collaboration

PoS(Lattice 2011)031

1. AuroraScience basic facts and goals

AuroraScience[1] is a project at the crossroads of Computational Sciences and Computer Architecture. It officially started on July 31st 2009.

The goals were:

- the design of an effective High Performance Computing (HPC) architecture (Aurora) suitable to support scientific applications demanding high computing capabilities;
- the installation of a prototype demonstrating the effectiveness of the architecture and providing clear evidence of the scalability to a larger system;
- the exploitation of the prototype in a number of relevant scientific fields in which HPC is a key issue;
- the dissemination of technical skills and knowledge in HPC related areas.

Joining custom and commodity solutions was the fingerprint of the project. This could be achieved by setting up a joint Research and Development program with an industrial partner. A suitable industrial partner was found by signing a Letter of Intent with Eurotech SpA. The company had previous, qualified experience in collaborating with academic instututions in the field of HPC. Moreover, Eurotech had signed a Memorandum of Understanding over several years of technological collaboration with the Intel company, to cooperate on HPC developments.

AuroraScience is a joint-project of INFN (Istituto Nazionale di Fisica Nucleare) and PAT (Provincia Autonoma di Trento); the latter manages it through the Fondazione Bruno Kessler (FBK). It was devised to be made of two phases; the first phase was supposed to last up to 24 months and at the time of the Lattice conference the application for the second phase had been submitted since April 28th 2011. At the deadline at which the Conference proceedings are due, a final decision for the second phase has not yet been taken by INFN and FBK. In the meantime, work is nevertheless going on, with partial support (covering personnel hired on the project) having being extended till the end of the year.

2. The Aurora architecture

The overall architectural design of the Aurora system was based on previous experience of parallel machines for scientific applications (mainly APE machines, to which members of AuroraScience gave substantial contributions). The architectural features of Aurora are the following

- The basic building block is the Aurora **board**, hosting two Intel CPUs and having a fair amount of memory on-board (possible configurations are multiples of 6 GB). At the start of the project it was envisaged that more than one generation of processors would be available as the project would progress: first nodes were based on *Nehalem* processors; current technology is based on *Westmere* processors; we will receive the first nodes based on *Sandy-Bridge* in one month from the time at which these Conference Proceedings are written. The

distinctive feature of the board is its connectivity: on top of a 40-Gbit/s Infiniband adapter, it hosts 1 FPGA (Altera Stratix IV GX230) and 6 PMC-Sierra quad-link PHYs, enabling the deployment of the FPGA-based toroidal network firmware. Each board has its coldplate, ensuring liquid cooling.

- 16 boards make a **half-chassis**, together with a root-card (on the top) and a DC/DC trayer (on the bottom). All the elements are physically connected (data signals, power) by a backplane. Infiniband connectivity is provided via a 36-ports switch hosted in the root-card, which also controls power management and monitoring functionalities. The sub-chassis is the basic Aurora module, equipped with power supply, liquid cooling and connectivity (actually within a sub-chassis the toroidal network can close in up to two directions).

- Liquid cooling is an Aurora key feature. Without it, it would not be possible to ensure the high computing density which is an Aurora fingerprint. Energy consumption is strongly cut by this choice. It should be pointed out that the so-called *quick-disconnect* technology has been for the first time used in HPC: boards are grouped in groups of four, but each can be disconnected at any time, without halting the system. It should also be mentioned that at the FBK laboratory where the system is installed, due to an efficient design of the cooling infrastructure, we have free cooling whenever the air temperature is below 20°C.

- Two half-chassis sit back to back to form a **chassis**. This is the basic module on which one has full toroidal connectivity (we can close all three directions).

- Eight chassis make a **rack**, which could be replicated in large installations. One rack is the configuration we are aiming at deploying in the second phase, in addition to the present system.

Figure 1: The Aurora board, half-chassis and rack.

The final first phase prototype is built of three[1] fully populated chassis, for a total peak performance of 15 TFlops. The capability to overcome the problems that were inherent to the startup of

[1] plus one, actually; we still have an extra half-chassis, hosting the prototypal boards.

the project gave convincing evidence of the overall robustness of the design.

2.1 Aurora custom torus network

Aurora fingerprint is the union of high standard commodity structure with a custom network. The Aurora toroidal network is a dedicated porting of the **FTNW** project. FTNW is an FPGA-based implementation of a light-weight communication network to tightly interconnect commodity multi-core CPUs in a 3d torus topology; it is a project by M. Pivanti, F. Schifano and H. Simma[2, 3]. FTNW had been made available to the *QPACE* project before; in QPACE the *FlexIO* interface was provided by IBM (that machine is based on the *Cell* processor). In Aurora everything is developed within AuroraScience.

- The FTWN-project has been ported onto the Altera FPGA and the processor interface has been adapted to the Intel CPU.
- Loading configuration-firmware on the FPGA is easy, thus fully enhancing the FPGA capability to easily update the design.
- The driver and the low-level library have been adapted and optimized for Intel CPU instruction set.
- The FTNW communication protocol is a very light one, requiring no explicit synchronization between communicating processors. Namely (refere to Fig. 2)
  - Proc1 provides (red action) credit to NWP1
  - Data are (blue action) moved from Proc0 to NWP0
  - NWP0 (blue action) sends data to NWP1
  - NWP1 checks data for errors and (red action) sends back a ACK/NACK
  - NWP1 (blue action) moves data to Proc1
  - NWP1 (blue action) notifies Proc1 that data are available
- While all the cores on a board share the same network processor, a *virtual channels* mechanism enables efficient usage.

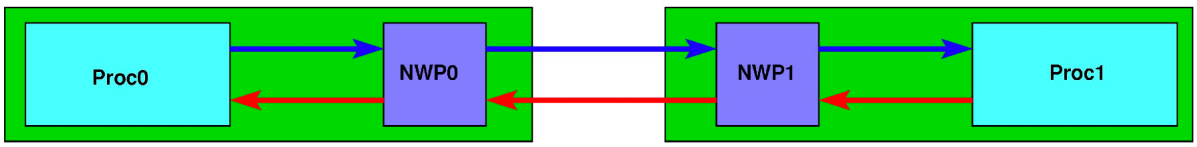

Figure 2: Basic FTNW communication steps.

2.2 Aurora environment for users

Aurora environment customizes standard tools: the system is thus quite user-friendly.

- The computing nodes are reached by four servers, providing a variety of services. The system has a partitioning mechanism: boards are grouped into partitions to exploit toroidal connectivity. A dedicated queue-system gives access to the resources. The queue-system provides facilities to best exploit both connectivity and the software layers that make the toroidal network available to programmers. We also have a mechanism to always fully exploit the resources: if for some reason some nodes can not group into a partition, they migrate to a *generic* queue whose resources do not guarantee toroidal connectivity (notice that Infiniband connectivity is in any case guaranteed).

- A NAS system based on 12+12 SAS disks (600GB faster disks and 2TB slower disks) provides both efficient runtime I/O and a fair amount of storage space.

- Compilers and MPI-environment are standard (*e.g.* gcc, icc, openMPI). At the programming level, users can use both low-level (more efficient) *atn* communication primitives and high-level *TORUS* and *torMPI* functions (the latter mimics MPI, while the former focuses on nearest neighbor communications). Communications software stack is sketched in Fig. 3.

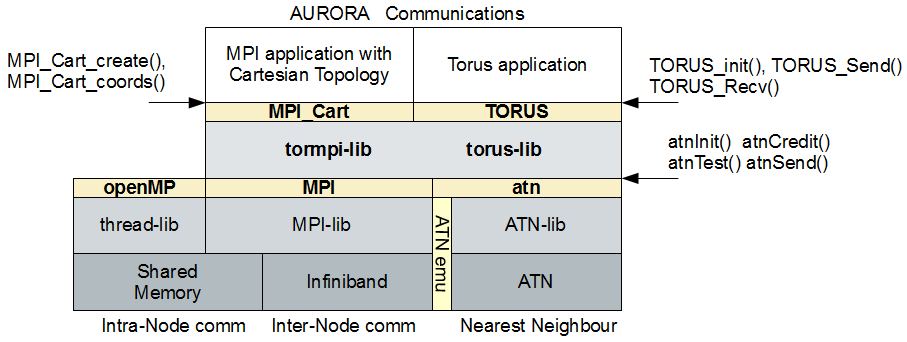

Figure 3: The communications software stack of Aurora.

3. Lattice QCD applications

To efficiently implement Lattice QCD applications on Aurora one has to face the basic challenge of balancing intra- and inter-node parallelism, as a direct consequence of the multi-core structure of the computing node. To have an idea of Lattice QCD codes performances on Aurora the interested reader is referred to another Lattice 2011 presentation[4].

3.1 IB and FTNW toghether in one application

Here we focus on a particualr issue: Aurora has two networks, and they can cooperate on a single task. We quote one application, to be regarded as a mere example, admittedly biased by the

scientific interests of the developers.

In the context of Numerical Stochastic Perturbation Theory[5], the order by order inversion of the Dirac operator is coded in the following formula (this is simply the order by order expansion of $\psi = M^{(-1)}\xi$, ξ being a source; ψ is expanded in perturbative orders because M - the Dirac operator - has an order by order expansion, while the source ξ has no expansion)

$$\begin{aligned}
\psi^{(0)} &= M^{(0)-1}\xi \\
\psi^{(1)} &= -M^{(0)-1}M^{(1)}\psi^{(0)} \\
\psi^{(2)} &= -M^{(0)-1}\left[M^{(2)}\psi^{(0)} + M^{(1)}\psi^{(1)}\right] \\
\psi^{(3)} &= -M^{(0)-1}\left[M^{(3)}\psi^{(0)} + M^{(2)}\psi^{(1)} + M^{(1)}\psi^{(2)}\right] \\
&\cdots \\
\psi^{(n)} &= -M^{(0)-1}\sum_{j=0}^{n-1} M^{(n-j)}\psi^{(j)}
\end{aligned}$$

- We notice that $M^{(0)-1}$ is diagonal in momentum space, while the $M^{(i)}$ are almost diagonal in configuration space. This suggests the strategy of going back and forth from momentum space via FFT: everything which is dealing with $M^{(i)}$ (what occurs in brackets) will be computed in configuration space, while we will perform an FFT before we apply $M^{(0)-1}$ and an inverse FFT to make the $\psi^{(i)}$ available to following orders computations;
- when one $\psi^{(i)}$ has been computed, we have the chance to advance in the computation of the (configuration space) brackets, *i.e.* computations in different space can overlap;
- the torus network is the natural candidate to perform communications needed to compute in configuration space (everything is almost diagonal, with only nearest neighbor contributions needed), while IB network can substain FFT.

This suggest the strategy which is sketched in Fig. 4:

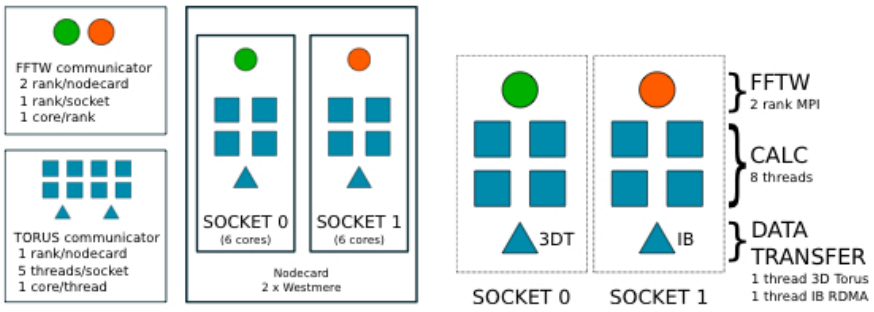

Figure 4: The computation/communication architecture of an application taking profit of both IB and custom network. Left: the general structure of the two communicators. Middle: the overall distribution of resources on a nodecard. Right: the matching of resources to tasks on a nodecard.

- The overall MPI application has two communicators (which we refer to as FFT and TORUS communicators).
- On each nodecard, we have three MPI processes, or ranks (if you take MPI jargon).
- Two ranks on each nodecard belong to the MPI communicator; they are single thread processes, residing on two physical cores (one per socket).
- The third rank on each nodecard is a multi-thread process, taking the ten residual cores available on the nodecard. Eight threads are actually in charge of computations; one core is in charge of communications on the torus network; one core is in charge of the (RDMA) MPI communications which make the FFT and TORUS communicators cross-exchange data.

We stress once again that this is a mere example. The two networks can cooperate and other applications can benefit as well by the same opportunity (an application asking at some point for a global scalar product is enough for providing another example).

Acknowledgments

The AuroraScience project is funded by the Provincia Autonoma di Trento and the Istituto Nazionale di Fisica Nucleare, in the framework of an agreement with the Fondazione Bruno Kessler. We wish to thank our collaborators at Eurotech and Intel. We are grateful to the members of QPACE and PetaQCD for stimulating discussions. Lattice QCD software developments and applications on Aurora are also supported by ITN STRONGnet (European Union Grant Agreement PITN-GA-2009-238353).

References

[1] L. Scorzato, *AuroraScience*, PoS (Lattice2010) 039.

[2] http://sourceforge.net/projects/ftnw

[3] M. Pivanti, F. Schifano and H. Simma, *An FPGA-based Torus Communication Network*, PoS (Lattice2010) 038.

[4] M. Brambilla, F. Di Renzo, M. Grossi, *Efficiency on multi-core CPUs: the Wilson Dirac operator on Aurora*, PoS (Lattice2011) 302.

[5] F. Di Renzo and L. Scorzato, *Numerical Stochastic Perturbation Theory for full QCD*, JHEP 0410 (2004) 073 [arXiv:hep-lat/0410010].