No. 182
May 2010

Bits & Bytes

RECHENZENTRUM GARCHING

Garching Computing Center of the Max Planck Society and the Institute for Plasma Physics*
Boltzmannstraße 2, D-85748 Garching bei München

New Bits & Bytes

Tilman Dannert, Markus Rampp

Dear reader,

after a longer break we are going to resume our semiannual schedule of the Bits & Bytes Computer Bulletin of the RZG. Starting with this issue, No 182, Bits & Bytes is published as a hyperlinked and 'living' document on the RZG web pages in the future http://www.rzg.mpg.de. A condensed PDF version will be distributed via e-mail to all registered users of our HPC systems.

At the same time we have opted to discontinue the shipping of hardcopies via regular mail. Besides environmental considerations, this allows for relaxed page-count limits and a more flexible design. Along with these changes and the introduction of a new layout readers may also notice that thematically the focus has somewhat shifted towards applications and application support. With the help of concise articles centered around practical examples we'd like to draw the attention of RZG users to relevant software and in particular to new tools, e.g. for debugging and performance analysis of HPC applications. The main aim is to 'lower the bar' for non-specialists, i.e. to allow users a quicker and more efficient evaluation of the relevance of a particular tool and to facilitate its adoption and efficient usage.

We hope that these updates can further improve the value of the Bits & Bytes bulletin for the researchers' practical work on RZG's computer systems. Comments and critics are welcome, please send it to T.Dannert@rzg.mpg.de.

Introducing Environment Modules

Markus Rampp

Background: Environment modules

RZG has introduced the environment modules (cf. http://modules.sourceforge.net/) concept for managing major software installations and for adapting the user environment on the main platforms, Linux (SLES) and IBM AIX. The well known modules system is meanwhile employed by many computing centres. The main purposes are 1) to adapt the user environment in order to allow the user to locate and transparently work with software installed at various locations in the file system, and 2) to switch between different versions resp. releases of the same software package. In particular, the user is no longer required to explicitly specify paths for different executable versions, or keep track of PATH, MANPATH and related environment variables in various shell-specific commands or profiles. Instead, users simply 'load' and 'unload' modules to control their shell environment. To this end, system administrators provide so-called modulefiles which are typically named by the software package and optionally a version number (e.g. xlf/12.0). The most popular shells are supported, including bash, ksh, and tcsh. Besides managing different software versions, the modules approach allows system administrators to install software in non-standard locations and also to relocate software packages in a way that is transparent for the user by adapting the modulefile.

Usage of modules

For the user the most important module commands are (complete reference at http://modules.sourceforge.net/):

module help lists module subcommands and switches

module avail lists available software packages and versions which can be enabled ('loaded') with the module command.

module apropos ⟨**keyword**⟩ searches available modulefiles for the specified keyword string and list all matching modules

*Tel.: +49(89) 3299-01, e-mail: benutzerberatung@rzg.mpg.de, URL: http://www.rzg.mpg.de/
Editor: T. Dannert, Max-Planck-Institut für Plasmaphysik, Boltzmannstr. 2, 85748 Garching

module help ⟨**package**⟩/⟨**version**⟩ provides brief documentation for the specified module.

module show ⟨**package**⟩/⟨**version**⟩ provides information about the impact of the specified module to the user's environment.

module load ⟨**package**⟩/⟨**version**⟩ 'loads' the module, i.e. modifies the user's environment ($PATH, $MANPATH, etc.)

module unload ⟨**package**⟩/⟨**version**⟩ 'unloads' the module

module list lists all modules which are currently loaded in the user's environment

Note that the module command is provided by default only in interactive login shells. For non-interactive shells (e.g. in shell scripts) a specific system profile must be explicitly sourced (see the RZG documentation). In addition to Linux (SLES) and IBM AIX, RZG maintains modulefiles (but no 'client configuration' of the module system) also for certain SUN SOLARIS software packages, which are centrally managed by RZG (e.g., matlab, NAG compilers and libraries, ...).

It is recommended to reference the environment variables set by the modulefile in scripts, makefiles etc., rather than of relying on absolute paths to libraries, binaries etc. By convention, an RZG modulefile sets an environment variable named ⟨PKG⟩_HOME (where PKG is the name of the package, for example: MKL_HOME) which points to the root directory of the installation path (see below for example usage). Information about additional, package-specific environment variables can be obtained with the commands module help ⟨package⟩/⟨version⟩ and module show ⟨package⟩/⟨version⟩.

Examples:

1) Interactive session on the command line, using the Intel fortran compiler (default version) and Intel MKL (version 10.2 explicitly specified):

```
~> module load intel
~> module load mkl/10.2
~> ifort -I$MKL_HOME/include example.F \
         -L$MKL_HOME/lib/em64t \
         -lmkl_intel_lp64 \
         -lmkl_sequential \
         -lmkl_core
```

2) Makefile (fragment):

```
FC=ifort

example:        example.F
        $(FC) -I$MKL_HOME/include test.F \
      -L$MKL_HOME/lib/em64t \
      -lmkl_intel_lp64 \
      -lmkl_sequential \
      -lmkl_core
```

GPGPU Computing

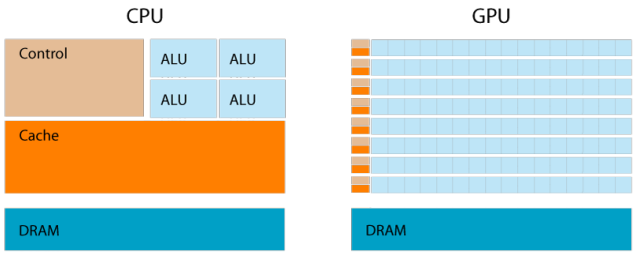
Klaus Reuter, Markus Rampp

Some background on GPGPU computing

Originally designed to be used in video cards, graphics processing units (GPUs) have come into vogue in recent years as cost and energy efficient highly-parallel platforms for general-purpose numerical computation. This so-called GPGPU (General Purpose Graphics Processing Unit) computing is now facilitated by AMD's ATI Stream and NVIDIA's CUDA software development platforms and runtime environments. So far, both technologies have attracted a large community of users and developers. For a first overview of resources, news and activities see the vendor-independent GPGPU.org portal or visit the websites of AMD and NVIDIA. The current evolution of GPU hardware technology clearly follows the trend towards utilizing and promoting GPUs for general purpose computations. For example, the next generation of NVIDIA cards (labeled 'Fermi') will offer error correcting (ECC) memory and hardware support for high-performance double-precision arithmetic. Both features are mostly irrelevant for the original purpose of GPUs (which is to accelerate the generation of computer graphics), but are often crucial in order to obtain a level of accuracy and reliability required for numerical simulations in science and engineering. The lack of ECC memory in the current NVIDIA 'Tesla' cards has in fact been observed to cause intolerable errors in some scientific applications. While the glossy promises of major GPU vendors and technology enthusiasts (reporting performance gains of up to several orders of magnitude) do not always hold for real-world applications, efficient GPU implementations can indeed outperform optimized CPU implementations by a factor of 10 or more when comparing state-of-the-art GPU and CPU platforms. Today, the growing portfolio of publicly available (often open source) GPU implementations of popular scientific application codes makes even moderate-sized GPGPU systems such as RZG's NVIDIA S1070 with 4 GPUs (see below) an attractive alternative to using, e.g., a small cluster of multicore CPU servers.

Differences between CPUs and GPUs

GPUs are different from CPUs in many respects. Most importantly, GPU devices are not operated as independent compute platforms but are attached to a conventional host CPU system. In that context, the GPU is often regarded as a 'coprocessor' or 'accelerator.' The limited bandwidth of the PCIe bus connecting the host with the devices puts constraints on numerical algorithms suitable for GPGPU computing. Data transfers from the host to the device have to be minimized, respectively balanced with the computational intensity of the calculations delegated to the GPU. As a rule of thumb reported by practitioners, the operation count of algorithms should scale at least with the square (N^2) or even cube (N^3) of the amount N of the transferred data.



On the GPU, SIMD methodology (single instruction, multiple data) applies. Consequently, data-parallel algorithms benefit most from GPGPU computing. Using one of the established software-development frameworks (such as ATI Stream, CUDA, or OpenCL), the programmer implements a multi-threaded code. During runtime, blocks of threads are generated which execute independently from each other. Comparing typical CPU and GPU designs on the hardware level, a larger fraction of transistors is devoted to data processing than to caching and flow control in a GPU, leading to a much larger number of arithmetic logic units (ALUs, see figure). For example, the current NVIDIA 'Tesla' GPU contains 30 so-called multiprocessors with 8 cores each. Moreover, a large number of threads is managed efficiently in hardware. Careful consideration of the memory layout and hierarchy is crucial to achieve good performance. Penalties for not taking into account the GPU architecture usually turn out to be much more severe than programmers' experience with multicore or vector processors would suggest. The large number of ALUs motivates why GPUs are sometimes classified as 'manycore' processors, alluding to the multicore architecture of CPUs and anticipating a possible convergence of both technologies in the future.

The OpenCL framework

Obvious concerns about the lack of portability of GPGPU applications across different platforms can be somewhat appeased by considering the achievements of recent efforts in standardization. The major players, including the competitors AMD and NVIDIA, have partnered to create, maintain, and actively develop OpenCL, which is an open standard for writing portable programs for heterogeneous platforms consisting of GPUs, CPUs, and other kinds of processing units. While the proprietary software platforms may currently still offer superior ease-of-development and application performance, OpenCL clearly provides a promising perspective for developing and managing portable GPGPU applications. Applications developed under CUDA today may be translated to OpenCL at a later time by employing a more or less automatic procedure, as NVIDIA claims.

The NVIDIA Tesla S1070 platform deployed at RZG

In December 2009, RZG has installed a system for developing and testing GPGPU computing applications and for evaluating the potential of GPUs for high-performance computing. The system comprises a compact NVIDIA Tesla S1070 unit with four FX 5800 GPUs and two CPU servers, each equipped with two Intel 'Nehalem' quad-cores. Each CPU server is attached to 2 GPUs of the Tesla unit. The software environment supports application development using the NVIDIA CUDA toolkit or OpenCL. A detailed and up-to-date technical documentation can be found on the RZG web pages.

The system is mainly dedicated to testing, benchmarking and developing GPGPU applications by or in collaboration with the application support group at RZG. Production applications can be hosted and scheduled on request. A number of popular scientific codes, mostly originating from the bioinformatics (blastp, smith-waterman, hmmer, ...), molecular dynamics, and quantum chemistry (gromacs, lammps, ...) domains have already been ported to GPUs and are publicly available at the NVIDIA website.

Users interested in developing or running GPGPU applications are kindly asked to contact RZG application support (contact: markus.rampp@rzg.mpg.de).

Applications: Performance Analysis with Scalasca

Tilman Dannert

Short description

Since some months a new performance analysis tool is available at RZG. The scalasca tool is developped by the Forschungszentrum Jülich and is able to automatically instrument a code on subroutine level or one can also instrument the code by user regions. The tool is designed to be scalable up to tens of thousands of cores and even

more. The main perspective of scalasca is the MPI performance, it is able to measure wait times, load imbalances and synchronization times in the trace mode.

It is available on vip, genius and on the Linux clusters via the loading of the respective module.

Example usage

We will show the usage of the tool by examining the user code GENE on Power6. After logging into the machine we load the scalasca module:

```
module load scalasca/1.3
```

Compilation

Now we have to recompile the source code with the scalasca instrumenter. This is simply done by prefixing the compile and link commands with

```
scalasca -instrument
```

This is easiest done by manipulating the makefile. In our case the Fortran compiler is set by the makefile variable FC, which is now prefixed and gives

```
FC = scalasca -instrument mpxlf95_r
```

Execution

After compiling, we can run the instrumented binary with the scalasca analyzer. To do so, we prefix the usual MPI runcommand with

```
scalasca -analyze
```

This gives for our example configuration with 8 processors the command

```
scalasca -analyze poe ./gene -procs 8
```

This starts by default the summary mode (instead of trace mode) of scalasca. It creates a directory and therein a file summary.cube.gz. The directory which is created to hold the summary or trace data is created by scalasca with a name constructed of the code name, number of processors etc. But the name can also be given by the EPK_TITLE environment variable. It is necessary to set this variable to a non-existing name as otherwise scalasca will abort.

Inspection of the results

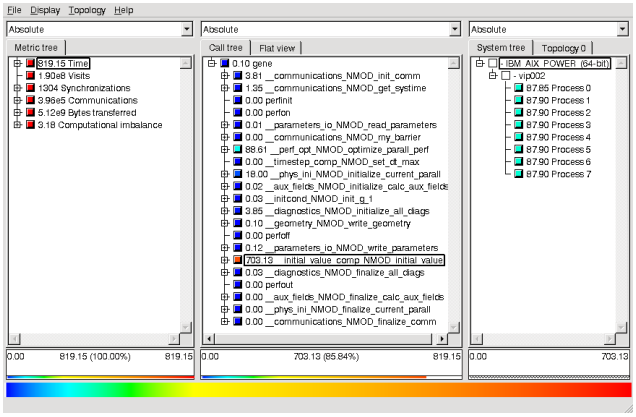
To read the performance results, enter the third call of scalasca by

```
scalasca -examine subdir/summary.cube.gz
```

This starts the CUBE graphical interface for the performance data. The description of this interface can be found in the User guide for scalasca. Roughly described, the cube interface consist of three views:

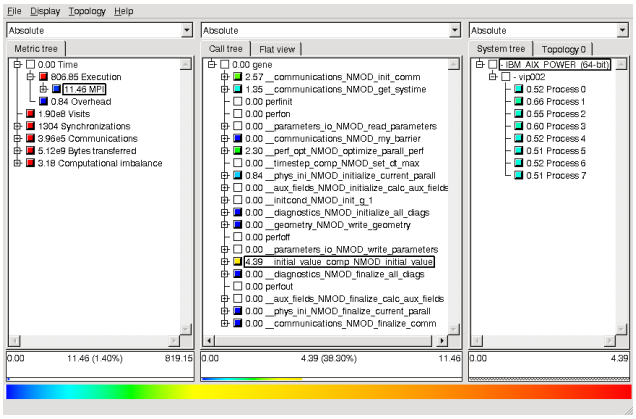
1. Left are the metrices of the summary, like time, or time in communications etc.
2. In the middle, you find the source related things, like subroutine name or name of user region in a call tree.
3. The rightmost window shows the system tree, i.e. the nodes on which the code ran. This is good for controlling load balance.

For the example, we get the following CUBE view:



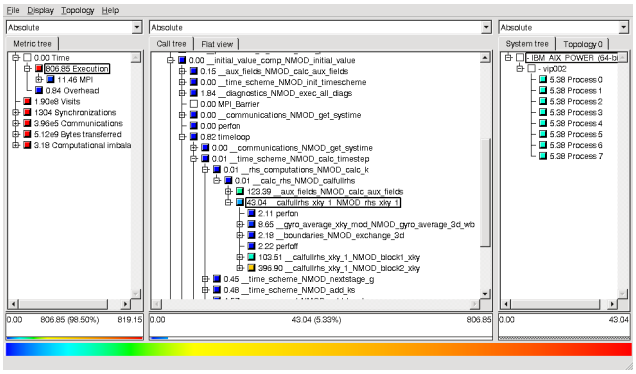
What we can learn from this view is that the total time of the run was 819 s, from which 703 were spent in the subroutine __initial_value_comp_NMOD_initial_value and these 703 seconds are equally distributed over the 8 processes, each taking 87.90 s.

To get more information, one has to dig further into the call tree and has to investigate several different metrices. One common use case is to distinguish compute and communication time. If all communication is done via MPI one gets this result by extending the metric tree.



Now one can clearly see that the communication only takes 11.5 seconds compared to 806 s of computation. For the timeloop of the code, which is hidden in the initial_value subroutine, one has only 4.39 s of communication, which will not be relevant for performance optimization. To further optimize the code, one has to look more for single-processor performance. To do this with hardware counters, one has first to further cook down the performance to some routines.

So we extend the call tree and get



From this view, one can deduce, that it would be advantageous to instrument the three routines

```
__aux_fields_NMOD_calc_aux_fields
__calfulrhs_xky_1_NMOD_block1_xky
__calfullrhs_xky_1_NMOD_block2_xky
```

for further investigation with performance counters to see, where one can improve performance. One possibility to use hardware counters is inside of scalasca by using the PAPI support of scalasca. Another possibility is to use the High performance toolkit (HPCT) from IBM on Power6 or BlueGene architecture. It will be topic of the next issue of Bits and Bytes, but documentation is already accessible at http://www.rzg.mpg.de/computing/hardware/Power6/profiling.html.

Network Configuration Templates for PCs

Christof Hanke

RZG now provides some templates of how to configure stand-alone PCs within the IPP to use NTP (time-server), Kerberos and AFS efficiently. For the locations Greifswald and Garching, separate templates have been prepared. If you are not within IPP, but want to use the AFS-cell 'ipp-garching.mpg.de' you should use the configuration for the location closest to you. If you are in doubt, use the location Garching. These templates are meant to be used for MS-Windows PCs which are not a member of the Active Domains (ipp.mpg.de, ipp-hgw.mpg.de) or UNIX-machines.

The configuration templates are located under : http://www.rzg.mpg.de/networkservices/configuring-your-pc

New RZG Gateway Machine

Ingeborg Weidl

In February 2009, the IBM Power4 hardware of the RZG gateway machine 'gate.rzg.mpg.de' ('sp.rzg.mpg.de') was replaced by the new Power6-550 technology. 'gate' provides access to the RZG computing facilities for general users. The machine has 8 CPUs available and 32 GB of main memory. The operation system is AIX 5.3. The users' home directories are located in AFS. Batch processing is no longer supported on 'gate' ('sp').

Update on Mass Storage

Manuel Panea-Doblado

The tape library used by the TSM backup and archive servers, a SUN/Stk SL8500 with 10000 tape slots, was moved last July to a new location in the computing centre. This was necessary in order to expand it, since the available floor space at the old location was not enough. The tape library is about 7 meters long and 2 meters wide. After it had been moved, a second library with 1500 tape slots was installed next to the first one. The two libraries are interconnected, allowing any tape to be mounted in any tape drive. In addition, 7 new LTO4 tape drives were installed, giving a total of 24 LTO3 and 21 LTO4 tape drives in use by the two libraries.

An analysis of the amount of stored data reveals an exponential growth of the total data volume over time, with a doubling of the total volume about every one and a half years. By the end of 2009, the second tape library will be further expanded to about 5000 tape slots. By mid-2010 the next generation LTO5 tapes -with a native capacity of 1.6 terabytes, i.e. double the capacity of LTO4 and four times that of LTO3 tapes- are expected to become available on the market. This will allow us to copy all our LTO3 tapes to LTO5 tapes, thereby freeing up several thousand

tape slots for new data.

Last spring, the computers where all our TSM server processes run were replaced by new machines. All servers (except two) run now on one of four identical IBM Power6 machines, each with 16 CPUs, 32 GB of RAM, Gigabit Ethernet, 4 Fibrechannel ports and about 8 terabyes of disk space. Of the remaining two servers, one runs on a node of the High Performace Computer, managing a Hierarchical Storage System to provide users with virtually unlimited storage space. The other one runs on a dedicated machine and keeps data sent by the TSM servers at the LRZ (Leibniz Rechenzentrum), with whom we have an agreement to cross-mirror certain categories of data, thereby increasing data availability and reliability.

System Environments for HPC

Ingeborg Weidl, Johannes Reetz

IBM Power6 system 'vip'

The IBM Power6 supercomputer 'vip' (207 compute nodes, 6 I/O nodes, 1 node for the Hierarchical Storage Management, all connected by a fast 8-plane InfiniBand network) is in production since June 2008. A Power6 node has 32 processors, each with 2 hardware threads, thus there are 6624 processors (or 13248 logical CPUs) available for computing, with a total main memory of 18.5 TB and a peak performance of 120 TF/s.

The operating system of the Power6 cluster is AIX 6.1 with the traditional parallel programming environment (MPI, ESSL/PESSL). The current compiler levels are Fortran xlf 12.1, C xlc 10.0 and C++ xlC 10.0. The batch system is LoadLeveler 3.5.

The Power6 processors can be used in 'Simultaneaous Multithreading' (SMT) mode. The SMT mode increases the performance of most applications significantly. We are using the SMT mode with 64 logical CPUs as the default on the Power6 nodes, but it is possible to use a Power6 node in 'Single Thread' (ST) mode with 32 CPUs as well. Meanwhile, about half of the batch jobs are using the SMT mode.

Testing and debugging programs interactively can be done on a dedicated Power6 node, the vip100. On the login node vip.rzg.mpg.de (vip001), interactive usage of poe is not allowed to avoid memory constraints.

The total disk space on the Power6 system is about 400 TB. There are 3 GPFS file systems available that are symmetrically accessible from all Power6 nodes:

/u (60 TB) for permanent user data. The users' home directories are in /u.

/ptmp (320 TB) for temporary job I/O. Files in /ptmp that have been not accessed for more than 14 days are removed automatically.

/r for migrated data (with an online disk space of 30 TB).

The AFS file system is available only on the login node vip001 and on vip100. Please note that no system backups are done neither of /u nor /ptmp.

Like the former Regatta cluster, the Power6 cluster 'vip' is part of the European DEISA (Distributed European Infrastructure for Supercomputing Applications) project. Within the 'DEISA Extreme Computing Initiative' (DECI) the RZG provides computing resources for the most challenging applications in Material Science, Bio Sciences, Plasma Physics, Earth Sciences, and Engineering.

Since the beginning, the Power6 machine is very well utilized (about 90-95%), and a considerable number of applications is using 512 and more processors.

IBM BlueGene/P genius

In October 2008, the Blue Gene/P system 'genius' was upgraded from 3 to 4 racks. There are now 4096 quad-core processors (i.e. 16384 cores) available with a total main memory of 8 TB. The peak performance is 55 TF/s.

Communication is done using a fast 3D torus network, disk I/O is done via a 10 Gbit/s Ethernet network. The GPFS file systems /u and /ptmp are shared by the Blue Gene/P and the Power6 system. AFS is only available on the login node 'genius.rzg.mpg.de', not on the Blue Gene/P compute nodes.

The operating system of the Blue Gene/P is Linux (SLES10), but the programming environment includes the Fortran xlf 11.1 and C/C++ 9.0 compilers and the ESSL library from IBM. As the batch system we are running LoadLeveler 3.5.

Most of the highly parallel applications on 'genius' are using 2048 cores up to 8192 cores. The utilization of the machine is around 85-90%. The Blue Gene/P also participates in the DEISA project.