

The OpenMP Source Code Repository: an Infrastructure to Contribute to the Development of OpenMP

Antonio J. Dorta, Casiano Rodríguez and Francisco de Sande
Dpto. de Estadística, I. O. y Computación
Universidad de La Laguna
38271 La Laguna, Spain
Email: {ajdorta, casiano, fsande}@ull.es

Arturo González-Escribano
Dpto. de Informática
Universidad de Valladolid
Spain
Email: arturo@infor.uva.es

Abstract

The OpenMP source code repository is based on a set of representative applications and it is supported by a web site. It is an infrastructure that we make available to the community of OpenMP users with the purpose of promoting discussions, and results interchange in the field of OpenMP research.

We want to appeal the OpenMP community to use the repository. Researchers may use it to publish the source codes developed and used for their experimental research and it is also a useful tool from the perspective of widespreading the OpenMP usage and knowledge.

The discussions originated by the use of this framework will define the real needs of the users and developers community, and they will lead the future of this work.

1 Introduction

Usually, when a research team publishes a work containing computational results, the corresponding source codes are not available to the scientific community. Upon request, authors usually provide a copy of their source codes, but even in this case, many details concerning to the experiment may still remain unrevealed.

In this work we present the OpenMP Source Code Repository (OmpSCR), a repository of applications parallelized using OpenMP. OmpSCR [1] is based on a set of OpenMP programs, but it is more than this: it is an open, non proprietary and freely accessible infrastructure oriented to share OpenMP programs with the aim of promoting discussions and results interchange in the field of OpenMP research.

Besides storing representative source codes written in OpenMP, OmpSCR provides some services that allow the OpenMP community to actively and dynamically discuss

about aspects related to these codes. We have created OmpSCR with the belief that this structure will be a useful tool for the future development of OpenMP, and that setting up this framework is a need if we want to widespread the use of OpenMP.

The remaining of the paper is organized as follows: In section 2 we deepen in the motivations of this work and we outline our goals. Section 3 is dedicated to explain aspects related to the source codes currently included in the repository. In section 4 we show the most relevant characteristics of the web site that supports the repository. Some computational results obtained for different platforms using the programs in the repository are presented in section 5. Finally, section 6 offers conclusions and comments on future work.

2 OmpSCR: the OpenMP source code repository

It is not always easy to know the details about how a computational experiment has been accomplished. Details like the way in which execution times have been measured, the points in the code where they have been measured, the arrangement of the input data for the application or the size of these data are some of the factors to be considered.

Lets consider [2] as an example. Among the computational results presented in this work the authors show the speedup obtained when sorting a 10^6 elements vector using the *Quicksort* algorithm. Reading the information provided in the paper about the experimental framework the reader can know the compiler, the platform, the number of processors involved or the speedups obtained. Nevertheless there are fundamental details that do not result clear. All these details are not specified when writing a scientific paper due to space limitations. However, it is crucial to know them to understand a work and to establish its relevance. A similar situation occurs in many other works where computational experiments are involved.

We believe that a first step to know the way in which experiments are performed is to have some mechanism that allows access to the source code. Furthermore, in order to compare results it is very advantageous to have source codes with a similar programming style, well documented, with similar execution interfaces and with an agreed methodology for performance measurement.

This is the main idea behind the creation of `OmpSCR`: to provide the OpenMP users with an infrastructure that allow both to evaluate the performance of OpenMP codes and to compare it across different platforms and compilers. It should be a framework that simplifies, as far as possible, the process of massive experimentation and results processing. It will rapidly allow to choose representative applications. The results obtained will be easily comparable with those coming from other groups.

With the recent arrival of OpenMP as *de facto* standard for shared memory parallel systems programming, the need of applications that allow to measure the performance has become clear. If recent works in this field are studied, we find that two different approaches have been used: initially the researchers used a limited set of applications to test their implementations [2]. In the recent time, this lack has been partially covered by the benchmarks developers providing OpenMP versions for existing benchmarks or developing specific new OpenMP benchmarks [3], [4], [5].

The most popular benchmarks in this field (SPEC and NAS) are highly oriented to a pure performance analysis and mostly they pay attention to numerical applications and to simple parallelization of loops. It is difficult to find in these benchmarks, for example, applications that use multilevel parallelism [6], a characteristic of the OpenMP standard widely discussed in OpenMP conferences. This is the motivation for many researchers that still continue using specific applications that make evident situations usually not considered in the classic benchmarks.

One of the goals of the `OmpSCR` is to standardize a set of applications to be an alternative to these classic benchmarks. The characteristics that the `OmpSCR` applications should fulfill are:

- They should be coded using ANSI C/C++ or standard Fortran90/95.
- Both kernel applications and complete applications can be considered to be included in the repository.
- In any case they should be programs reasonably simple (up to 5000 code lines). This will ease their use by non-expert OpenMP users.
- Applications will be representative of the real world, or applications that make evident situations that require special treatment when programmed using OpenMP.

- They should be codes extensively tested and validated in a wide variety of systems.
- The code must be written according to quality standards.
- The applications must be well documented.
- It is a desirable property that the application be self-checking (containing code to test the correctness of its results).
- The programs must be freely distributed, and not affected by copyrights.

The approach that we plan to follow in the development of the `OmpSCR` is collaborative and incremental. We hope that different research groups contribute their own codes to be added to the repository, as far as we know that many groups use codes compliant with the above characteristics. We think that the repository must be open, not limiting the number of applications available on it. Any contribution can be incorporated if it fits the former requisites.

Initially we do not plan the creation of an OpenMP benchmark, but we do not discard this future evolution.

3 Applications in the `OmpSCR`

The number of applications currently available in the `OmpSCR` is 10 but the `OmpSCR` team is actively working to add new applications. The number of programs available will clearly increase depending on the interest raised on the OpenMP users and developers community.

Although most of the applications in the repository are coded using C, the `OmpSCR` is not oriented to a specific programming language. One of the aims of `OmpSCR` is to make available the same code written in different languages to check the behavior of the corresponding compilers.

The kind of applications included in `OmpSCR` goes from very simple examples of OpenMP parallelizations to complex codes with several parallel loops. Some of the programs constitute examples for different parallelizations that are not usual or that can not be easily implemented using OpenMP.

It is also our goal to include applications with specific interest for OpenMP compiler designers. For instance, some applications requiring multilevel parallelism have been included. Also there is an example of parallel graph search using a master-slave paradigm. `OmpSCR` is also thought as a repository of examples useful for learning how to properly use OpenMP. Thus, some of the programs are provided in several versions using different parallelization strategies or primitives.

For simple codes showing different ways of using OpenMP primitives it is even possible to include in the

repository bad-parallelized examples. For instance, we have included some examples of loops with carried dependences. The bad solutions degenerate in sequential codes or produce inconsistent results. These implementations are clearly marked as wrong in the distribution.

OmpSCR is distributed as a whole in a *tarball* archive. It contains a directory structure which may be installed in a local account. The distribution contains simple installation instructions, a guide for new applications developers and a FAQ file.

Inside the distribution, the directory `applications` contains specific subdirectories for each program or set of versions. In the download section of the OmpSCR web site, the applications are also available individually to be installed under the `applications` directory.

The only platform dependent information needed to compile the OmpSCR applications is the name of your OpenMP compilers front-ends and the appropriate command-line options for them. The final users must configure these details editing a configuration file or using an interactive script. Templates with usual options for common compilers and development platforms are also provided in the distribution.

To avoid vendor-specific make utility details the applications building is controlled with GNU `make`. In each application directory, its developer provides a very simple `GNUmakefile`. A template for this file is also provided in the distribution to be adapted by the new application developers. Usually, they only need to include the target binary name in a variable. At compile time, this standard `GNUmakefile` includes a common file with all the implicit rules needed to build C or Fortran applications, using the compiler details provided during the user's configuration. Thus, an application may be rebuild alone from its own directory, or with the whole distribution.

The programs written using each programming language are totally independent. If there is only one compiler definition, the compilation rules simply ignore the source code files for which there is no compiler definition. The binaries for the applications are stored in the `bin` directory of the distribution and command lines used to compile each application are also stored in a `log` directory for future reference.

All the applications in OmpSCR present a common style. All use the functionalities provided by a common module included in the distribution (stored in the `common` directory). This common toolkit-module unifies aspects related to the applications programmer interface. The module eases the automation of experimentation and results collection. It implements APIs for: command line arguments processing, timers definition, time measurements and execution reports generation. This module has bindings for C and Fortran90/95 languages, presenting a very similar API.

All the applications currently included in the OmpSCR distribution contains the GNU general public license. Although the OmpSCR team encourages the use of an open software license, the original author may protect her code as she wants, as far as she grants: free distribution of the source code in OmpSCR, free use of the generated program and free publication of performance results on any platform. Our purpose is to open the possibility for industry programmers to show up their developments and include real or even commercial applications in the repository.

The applications currently included in the repository are:

- Computing π
- Mandelbrot set area
- Molecular dynamic
- Quicksort
- Divide and conquer fast Fourier transform
- Bailey's "6-step" FFT
- Loops with dependences
- Cellular automaton
- LU decomposition
- Graph search

All of them are very well known applications. We have simply selected those programs that we have used in the past for our experiments. We have been also guided by the intention of including some diversity in the codes: there are synthetic codes (Loops with dependences) and real-life ones, there are simple codes (computing π) and more complex examples; there are numerical (LU decomposition) and non-numerical codes.

4 The OmpSCR infrastructure

The infrastructure of the OmpSCR web site [1] is based on PostNuke [7]. PostNuke is a portable multilingual open software content management system (CMS) written in PHP. It is specifically oriented to user communities in which users actively post contents using web browsers. We have chosen this type of tool to support OmpSCR because it simplifies the administration of such a web site. The CMS allows administrators to dynamically work with a structured environment, manipulating any kind of contents: articles, news, FAQs, file downloads, etc. With respect to a version control system, a CMS has the advantage of additional functionalities like discussion forums or interest links.

To join the community, a new user must register herself on the web site. This allows her to actively participate, posting contents, asking questions, or joining the discussion forums. Moreover, a registered user may configure her account, adapting her web site interface. Contributions from registered users must be reviewed and approved by the administrators before they are released.

We now quickly describe the most relevant modules of the web site, focusing on their practical use. However, the best information source about the web site contents and use can be found inside itself. The user will find a complete FAQ file with answers to most of the usual questions.

The *downloads* section is the core of *OmpSCR*. It allows to download the repository and other related information. Registered users may also apply to add new files to be *download* section filling up a simple form. This section has the following subsections: *Documentation*, *OmpSCR distribution*, *Publications*, and *Results*. The *OmpSCR* subsection contains the repository directory structure and applications, which may be downloaded separately. Results subsection includes performance results for the different applications measured for specific platforms. These results are shown as plots as well as raw numerical data. More results may be also supplied by users.

A registered user may also post other information items to the *OmpSCR* web site: news items, reviews, comments, or hyperlinks to interesting web sites. News are the easiest way to supply new information of any nature to the *OmpSCR* community. Usually the *news* section will be used to discuss new applications added to the repository, computational results, call for papers for related conferences, etc. Reviews and discussion forums are other alternatives to actively participate in *OmpSCR*.

5 Computational results

In this section we present some performance results obtained for some of the *OmpSCR* applications. We focus on the kind of discussions that such results may motivate, instead that on the values themselves.

Fig. 1 shows the results obtained when executing the *Mandelbrot* and *Molecular dynamics* applications on four different platforms. The Intel label corresponds to a shared-memory Intel-Xeon multiprocessor PC, with four 1.4 GHz processors, and a 400 MHz system-bus. The IBM tag represents an RS-6000 IBM machine with 375 Mhz Power3 processors with 64Gb of memory. HP corresponds to a Compaq HPC 320 with 1 GHz Alpha EV68 processors and 80Gb of memory. The SGI plot corresponds to a SGI Origin 3000 with 600 MHz MIPS R14000 processors, and 160Gb of memory. In all the platforms we have used the native OpenMP compiler. In the case of the Intel and IBM platforms we had exclusive use of the resources.

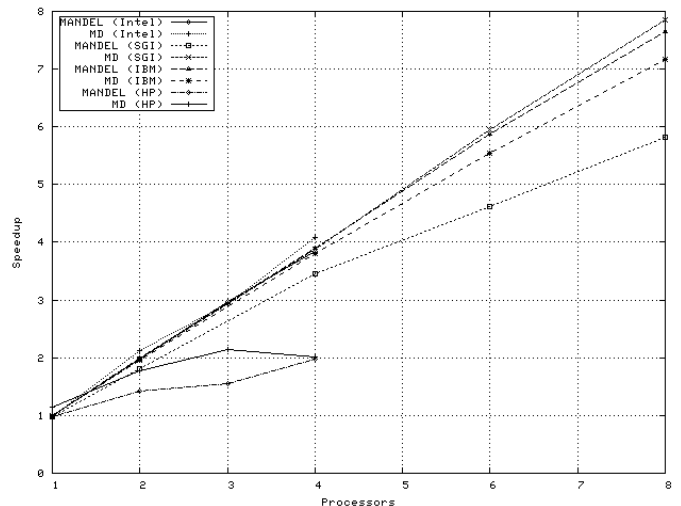


Figure 1. Results for Mandelbrot and MD applications

Both applications, Mandelbrot and Molecular dynamics, show a similar behavior for all the platforms: almost linear scalability, as expected. However, in the case of the Compaq platform, the non-exclusive use of the CPUs produces a negative impact on measures.

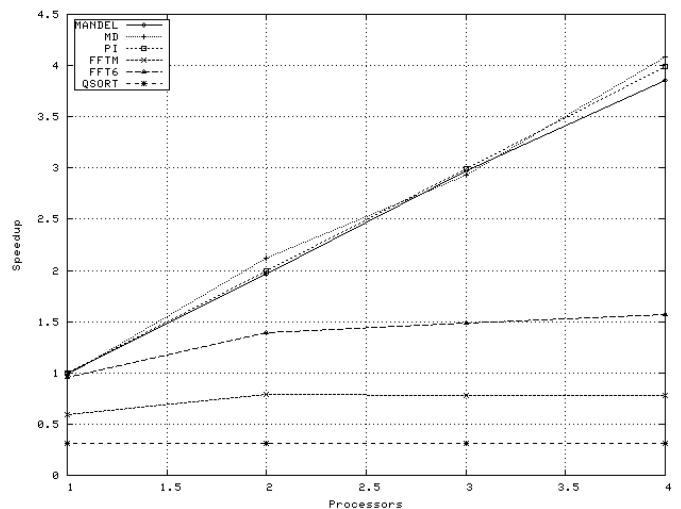


Figure 2. Results in the Intel based PC

On the other hand, Fig. 2 compares the speedups obtained on the same platform (Intel) for different applications. They are compiled using the Intel C++ 7.0 compiler for Linux. Applications considered are:

- *Mandelbrot* (MANTEL) with an input-data of 16k.
- *Molecular dynamics* (MD) with an input-data of 8k par-

ticles and 10 simulation steps.

- *Pi* (π), computed with a precision of 10^{-8} .
- *FFT* (FFT), with an input signal of 2M complex data-items.
- *FFT-Bailey* (FFT6), with an input signal of 1M complex data-items and 10 iterations.
- *Quicksort* (QSORT), sorting 4M integer data-items.

The first three applications show an almost linear scalability behavior. However, *FFT-Bailey* show a logarithmic speedup, and the last two applications sublinear speedups. This effect appears because the compiler is not exploiting the multilevel parallelism present in these applications.

6 Conclusions and future work

In this paper we have introduced *OmpSCR*, an infrastructure which purpose is to be a forum for the OpenMP users and developers community. It allows:

- To show and to evaluate different parallelization schemes for a given application.
- To evaluate and publish performance results of OpenMP codes on different platforms, using different languages or compilers.
- To provide free access to source codes and results published on the repository.
- Researchers may use it to publish the source codes developed and used for their experimental research.
- It may also allow the OpenMP community to discuss about new parallel constructs and mechanisms to be included in the language, showing the effect of their proposals on real codes.
- Finally, it will be used as a resource to help in the OpenMP learning, presenting real implementation examples of the language features.

For these reasons, we want to appeal the OpenMP community to use *OmpSCR*, and we offer our support to adapt those relevant applications whose developers are interested on including them in the repository.

We have presented the current maturity-level of *OmpSCR*, a project in active development. We want to remark some of the tasks in deployment stage:

- Popularization of *OmpSCR* among the OpenMP users and developers community.

- Finding and including new applications in the repository.
- To appeal to the active working groups on this field to contribute with their own codes.
- Conducting computational experiments on different platforms, and publishing the results in the web site.

Although *OmpSCR* is not proposed as an OpenMP benchmark on its own, we do not exclude the possibility of a further development in this direction. The acceptance and contribution of the OpenMP community will determine the long term scope of *OmpSCR*. The development of a complete benchmark implies some tasks to do:

- All the benchmark applications should contain code for automatic correctness check.
- Applications should print at the end of their execution a common and complete report of performance.
- Appropriate input data characteristics and sizes must be clearly stated for each application.
- A protocol for the communication of the benchmark results should be devised.

These tasks do not present special difficulties. However, we have preferred to introduce *OmpSCR* as a repository instead of a benchmark. The discussions originated by the use of this infrastructure will define the real needs of the users and developers community, and they will lead our future works.

Acknowledgments

Computational results have been obtained using the installations of CEPBA, CESGA and CIEMAT. This work has been partially supported by the Canary Islands government, contract PI2003/113, and also by the EC (FEDER) and the Spanish MCyT (Plan Nacional de I+D+I, contracts TIC2002-04498-C05-05 and TIC2002-04400-C03-03).

References

- [1] *OmpSCR OpenMP Source Code Repository*. <http://www.pcg.u11.es/ompscr/>.
- [2] Xinmin Tian, Milind Girkar, Sanjiv Shah, Douglas Armstrong, Ernesto Su, and Paul Petersen. Compiler support of the workqueuing execution model for intel smp architectures. In *Proc. of the Eighth International Workshop on High-Level Parallel Programming Models and Supportive Environments*, pages 47–55, Nice, Paris, Sep 2003.

- [3] H. Jin, M. Frumkin, and J. Yan. The OpenMP implementation of NAS parallel benchmarks and its performance. Technical Report NAS-99-011, NASA Ames Research Center, October 1999.
- [4] Vishal Aslot, Max Domeika, Rudolf Eigenmann, Greg Gaertner, Wesley B. Jones, and Bodo Parady. SPEComp: A new benchmark suite for measuring parallel computer performance. In *Proc. of of WOMPAT 2001, Workshop on OpenMP Applications and Tools*, volume 2104 of *LNCS*, pages 1–10, West Lafayette, IN, USA, July 2001.
- [5] Mitsuhsisa Sato, Kazuhiro Kusano, and Sigehisa Satoh. OpenMp benchmark using PARKBENCH. In *Proc. of the 2nd. European Workshop on OpenMP EWOMP2000*, Scotland, UK, September 2000.
- [6] Antonio J. Dorta, Jesús A. González, Casiano Rodríguez, and Francisco de Sande. Towards structured parallel programming. In *Proc. Fourth European Workshop on OpenMP (EWOMP 2002)*, Rome, Italy, Sep 2002.
- [7] PostNuke. <http://www.postnuke.com/>.