Automatic X-Kaapi peformance evaluation

Benjamin Briot —François Broquedis —Thierry Gautier

MOAIS
July 10, 2012

1 Software description

X-Kaapi is both a programming model and a runtime for high performance parallelism targeting multicore and distributed architectures. X-Kaapi was developed in the MOAIS INRIA project by Thierry Gautier, Fabien Le Mentec, and François Broquedis for the GCC/OpenMP runtime support.

In this report we describe the software architecture and requirements on measuring performances of a X-Kaapi release.

The runtime library also comes with a full set of complementary programming interfaces: C, C++, and STL-like interfaces. In this report, we described the benchmark suite and evaluation methodology for collecting performances across the BOTS and RODINIA standard OpenMP benchmarks using the X-Kaapi's libKOMP runtime which are ABI compliant to GCC/OpenMP libGOMP runtime.

X-Kaapi Contacts

If you wish to contact the XKaapi team, please visite the web site at:

`http://kaapi.gforge.inria.fr`

1.1 Overview of the software

X-Kaapi is available for download on the INRIA forge at `http://kaapi.gforge.inria.fr`. The source repository is managed by `git`. The master branch is composed of the following features:

runtime this is the low level part of X-Kaapi that allows to schedule in a portable fashion programs written as a sequence of tasks with dependencies (data flow dependencies). The scheduling algorithms are based on work stealing. The task model is based on the malleable task model with dependencies. From the data flow dependencies, a task becomes ready when all inputs are produced by previous writer tasks. Nevertheless, a running task may generate extra parallelism (if the user description gives the method for that).

api on top of the runtime level, few APIs are proposed to the users:

- API C: this a simplified C api to describe dependent tasks as well as independent loop such as parallelized by OpenMP.
- API C++: it is a more complete API based on template meta-programming, to write programs where tasks are abstract objects with several implementations (CPU, GPU -CUDA-).
- API F: a fortran API very closed to the C API provided by the API C level.

- libKOMP: a ABI compatible library with the GCC libGOMP runtime for OpenMP-3.0.
- libQuark: a ABI compatible with the Quark API used by the PLASMA project for linear algebra subroutines.

In this document, we focus on the libKOMP library that allows to execute for free GCC/OpenMP program on top of X-Kaapi. As presented in [1] the gain was better performance on fine grain OpenMP tasks.

1.2 Goals of automatic performance tracing tool

The goals of automatic performance tracing are:

- it must provide better understanding on runtime modification, such as modifying the work-stealing heuristic coded in X-Kaapi. Impact of modification will be experimented and measured on standard benchmark.
- it must provide a historical evolution of X-Kaapi library. By keeping in data bases performances of benchmarks and by the capability of comparing two same benchmarks it is possible to identify automatically which commit introduces performance difference.

2 The benchmark suites BOTS and RODINIA

BOTS and RODINIA are standard benchmark suites for OpenMP. We target them in order to provide up-to-date X-Kaapi implementation of libKOMP.

Each of these benchmark is composed of set of applications. Each application required specific inputs that are given as environment variable and/or command line options.

2.1 BOTS

BOTS is a benchmark suite dedicated to OpenMP-3.0 tasks. The reference paper is `http://capinfo.e.ac.upc.edu/PDFs/dir04/file003629.pdf`.

BOTS benchmarks can be download at `https://pm.bsc.es/projects/bots`.

We have developed a set of script to launch each application of the benchmark suite with some standard parameters. The scripts pass to the application the correct inputs and environment variables required to run our libKOMP runtime.

On output we collect time to run each application. The important inputs was:

- the input file for the benchmark or the set of parameter. It is fixed.

- the number of cores on which the application runs.
- the XKaapi scheduling policy or other parameters we want to test.

The output file contains the results of *Niter* runs of the applications with a given set of parameters. After that a ad-hoc tools collect timings to produce the average, standard deviation of each benchmark and for each number of core.

2.2 RODINIA

RODINIA is a benchmark suite of the Berkley's DWARF problems. Parallelization is provided on top of CUDA (and me be OpenCL) and OpenMP. OpenMP parallelization relies on the parallel (independent) loops annotated by `#pragma omp parallel loop`.

We modify some of the benchmarks in order to report time of the parallel region.

As for BOTS, a set of script is going to be developed by B. Briot. The output files will have the same structure as for the BOTS in order to reuse one of our scripts.

3 Benchmarking X-Kaapi with BOTS and RODINIA

Our goal is to report the performances of these benchmarks when we vary important features for parallel execution. Moreover, X-Kaapi is managed by git, thus it is important to report evolution of performances for different version of the library (each version is uniquely identify by its git hash number).

- the scheduling policy (default or specific scheduling strategy)
- the number of cores
- the way the library is configured, for instance the following configurations are often tested (from the most used to the less used).
  - default configuration is the "release configuration".
  - With options to take into account NUMA characteristic of the architecture.
  - With option to capture of some performance counters: the library add extra code to collect a set of software and hardware performance counters. The purpose here is to measure the overhead with respect to the release configuration.

No script yet exists to get the version of the library (git number), a configuration and make installation in order to launch benchmark suites.

On output we are interesting in following raw results for each benchmark of the suites:

- time in function of the number of cores
- speedup

From them, it would be better to build more interesting metrics to compare variation of a given configuration with respect to the reference configuration:

- classification of applications with respect to: 1/ conservation of the performances; 2/ slowdown in performances; 3/ gain in performances.
- synthetic metric such as overall gain or slowdown in %.

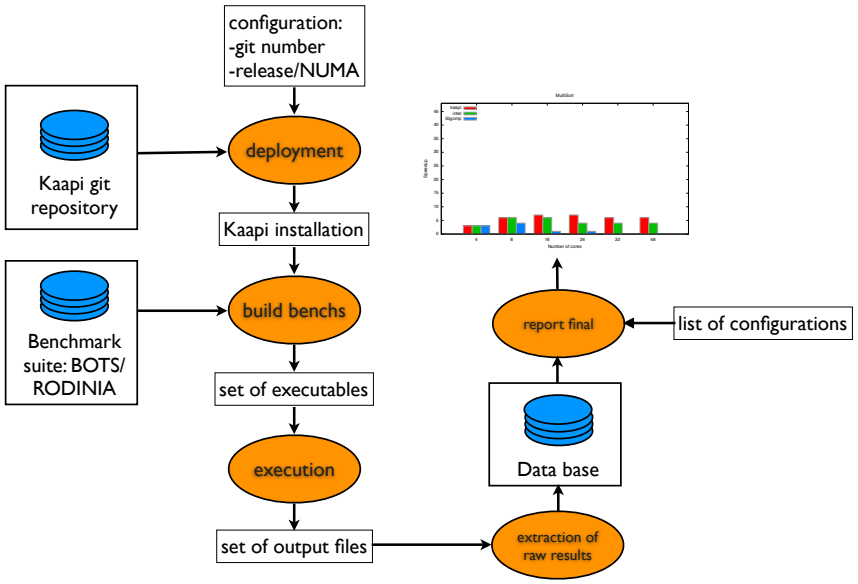

Figure 1: Global workflow of automatic benchmarking of X-Kaapi

References

[1] Francois Broquedis, Thierry Gautier, and Vincent Danjean. libKOMP, an Efficient OpenMP Runtime System for Both Fork-Join and Data Flow Paradigms. In *IWOMP*, pages 102–115, Rome, Italy, 2012.