The Open TS dynamic parallelization system approach

Sergei M. Abramov[1], Alexander Moskovsky[1,2], Vladimir Roganov[1] , Elena Shevchuk[1]

[1] Program Systems Institute, Russian Academy of Science,
52020 Yaroslavl region, Russia, Pereslavl Zalessky,
m. Botik, Progam Systems Institute, Russian Academy of Science
{abram,shev}@botik.ru,var@skif.botik.ru
http://www.botik.ru/~abram/
[2] Moscow State University, Chemistry Department,
119992 Moscow, Russian Federation, Vorobyevy Gory 1/3
moskov@lcc.chem.msu.ru

Abstract. The paper describes the Open TS – a system for parallel computing, which primary goal is to provide high-level programming tool for a wide variety of modern parallel computers. The C++ language has been extended with a few additional keywords that provide means to denote potential parallelism grains in a source code. A runtime system has been implemented on top of Message-Passing Interface (MPI), with dynamic load balancing. While overhead introduced by the Open TS runtime seems tolerable, the by-product benefits of implicit, dataflow-style approach to parallelizing computation are described: portability, fault-tolerance and adaptation to wide-area networks by synthesizing computational web-services.

1 Introduction

The Open TS systems is being developed by our group during the recent years. During this time, a number of applied programs and scientific codes has been developed with the help of this technology, see for instance [1],[2]. The primary development goal was to provide high-level tool for programming for computational clusters and SMP systems, however, an approach allows us to easily extend the system to grids of computational clusters as well.

The manuscript is organized as the following. In the first section, we outline the approach for writing parallel programs and we discuss the benefits of the Open TS approach: easy portability across many hardware platforms, fault-tolerance. The last one is dedicated to the benchmarks, including the re-implementation of ALCMD and MPI PovRay codes with Open TS technology.

2 Open TS approach.

The approach of Open T-system (which stands for the Open TS) is similar to the coarse –grain dataflow computational model, with some notable distinctions. The T++ language is devised, which is an extension of C++. In T++ programmer use a few additional keywords to designate what function invocations can produce grains of parallelism (dataflow graph nodes), what data can be shared by or transferred between the grains of parallelism (dataflow variables or "futures" [3], which called or "non-ready variables" in Open TS). The grains (both data and computation) should be kept large enough to make small the overhead of the runtime system.

The T++ is a "seamless" extension of C++: most T++ programs can compiled with C++ compiler using some "dumb" macro-definitions for T++ keywords, and result is a correct sequential program.

The two basic notions of the T++ are:

- T-function, which means pure functions with no side-effects, which can be used as a grain of parallelism. Open TS runtime can execute T-functions as independent threads, or migrate threads between computational cluster nodes, enabling load-balancing.
- T-variable. The variable can be cast to the "original" C++-type variable, which makes the thread of execution suspend until the value becomes ready. That it very similar to "dataflow variable", or "futures" [2]. That also differs T++ from "standard" data-flow models, where task is ready for execution only after all incoming data are ready – in opposite, threads in Open TS can be launched before any incoming data for a grain are ready.

The sample program, calculating Fibonacci numbers is introduced below:

```
tfun int fib(int n) {

          if (n<2) return 1;

          return (fib(n-1)+fib(n-2));

}

tfun int main (int argc, char *argv[]) {

          int n = atoi(argv[1]);

          printf("Fibonacci %d is %d\n",n,(int)fib(n));

          return 0;
}
```

The differences with the original C code are shown in bold: two “tfun” keywords and casting of non-ready value returned by the T-function “fib” to the original “int” type, which makes the “main” function thread to wait for the “fib” result.

Open TS provides very important features, which ease burden of application development and allows programmer concentrate on algorithm, not managing environment:

- Automatic garbage collection of non-used values
- Multiple assignments of dataflow variables.

To implement the last feature, a tricky protocol has been introduced, which describes the relationships between producer’s execution thread lifecycle and value “readiness” or availability to consumer’s threads.

Alternative technologies of compilation have been investigated: converter based on Open C++ [4] and front-end language for GNU compiler collection, the first solution is more flexible, while the last supports many advanced C++ language features.

The runtime support library is implemented in C++ and utilizes MPI subset for data exchange in a computational cluster environment.

An important implementation feature is a work migration model. In cluster environment, each MPI process has it's own set of tasks and it’s own meta-scheduler instance (thread) running. Tasks are T-function invocations (with some exceptions: sometimes it may be more efficient to evaluate T-function call immediately. So, T-functions are potential parallelism grains, not obligatory). Tasks can be either running or pre-natal. In a general case, pre-natal tasks can be moved between nodes, while running tasks cannot. Macro-scheduler can either send one or more pre-natal tasks o another node(s) or execute pre-natal task, thus making task running. When executing, task can invoke T-functions, thus creating more tasks (possibly, of another type). Optimal load balancing is achieved with the help of heuristic algorithms or algorithms, specific to certain applications.

3. Benefits of Open TS approach

The implicit parallelization gives the Open TS unique opportunity to adapt application to the computational resources available. Theoretically, some T++ programs can be compiled for such targets like FPGA, or FPGA+CPU devices. More practically, a load balancing heuristic can handle the situation, when CPU of different speeds are used in the computational cluster – even re-linking is not necessary. The Open TS applications can be run in Grid environment – using appropriate MPI implementation. However, load balancing techniques for grids is a subject of ongoing research.

The Open TS “T-functions” can be used to create computational web-services, and the appropriate tools have been developed by our group. That gives two opportunities:

- Integration of Open TS application to the larger computational environment (e.g. workflow) with help of XML-based tools.
- Work migration between computational clusters via web-service interface. One can imagine a set of computational clusters connected to the Internet, exchanging workload (T-tasks) via web-service interface.

Another important benefits of Open TS dataflow model is the ability to re-start tasks, which execution has failed for some reasons (e.g. due to hardware failure), similarly to other dataflow-based tools like Mentat [5]. In the Open TS, this is a bit more complicated, since software distributed shared memory is used to facilitate non-ready values implementation, so it's practical to implement "strict" fault-tolerant mode for T-applications.

3 Benchmarks.

The OpenTS implementation of embarrassingly parallel (EP) test of NASA NPB suite demonstrated 96% of theoretical speedup in a computational cluster with less than 10 nodes (EP class A) and up to 86% in 32 CPU cluster of 16 dual-CPU nodes (EP class C). EP benchmarks were also run on heterogeneous clusters (of different CPU speeds) and demonstrated good speedup also.

Other benchmarks include re-implementation of MPI patch for PovRay ray-tracer and MP_Lite library (part of Ames Lab Classic Molecular Dynamics code, ALCMD). Execution times of MPI and Open TS applications compared on various clusters and differ from each other by 5-10% only, while the volume of source code is 10-7 times smaller for Open TS.

4 Acknowledgements

This work is supported by basic research grant from Russian Academy of Science program "High-performance computing systems on new principles of computational process organization" and basic research program of Presidium of Russian Academy of Science "Development of basics for implementation of distributed scientific informational-computational environment on GRID technologies", as well as Russian Foundation of Basic Research grant 05-07-08005-ofi_a.

As well, we thank Igor Zagorovsky, German Matveev, Alexandr Inyukhin, Alexandr Vodmomerov, Eugene Stepanov, Ilya Konev, Elena Shevchuk, Yuri Shevchuk, Alexei Adamovich, Philip Koryaka and others, who contributed to the implementation and development of Open TS and previous T-system versions.

References

1. Arslambekov R.M., Potemkin V.A., Guccione S. Parallel version of MultiGen for multi-conformational analysis of biological activity of compounds// XII International Conference CMMASS'2003, Book of abstracts ;
2. A. Kornev, " On globally stable dynamic processes" //Russian Journal of Numerical Analysis and Mathematical Modelling, Volume 17, No. 5, p 472
3. http://en.wikipedia.org/wiki/Future_%28programming%29
4. Chiba S. A "Metaobject Protocol for C++" , In Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), page 285-299, October 1995. http://www.csg.is.titech.ac.jp/~chiba/openc++.html
5. A. Nguyen-Tuong, A. S. Grimshaw , J. F. Karpovich "Fault Tolerance via Replication in Coarse Grain Data-Flow", http://www.cs.virginia.edu/~jfk3w/papers/ft-paper.ps