نحو اختبار البرمجيات فائقة السرعة

مقدم البحث

أحمد محمد أحمد الغامدي

المشرف

أ.د. فتحي البرعي عيسى

المستخلص

الحوسبة عالية الأداء أصبحت ذات أهمية في العديد من القطاعات العلمية والصناعية. التزايد المستمر في بناء حاسبات عملاقة ذات قدرة حسابية كبيرة أصبح ملحوظاً والحاسبات العملاقة (Exascale) ستكون متاحة خلال السنوات القليلة القادمة. وكنتيجة لذلك، يصبح بناء أنظمة ضخمة تعمل على التوازي أكثر أهمية لمواكبة التقنيات القادمة المتعلقة بأنظمة (Exascale) أيضاً أصبح استخدام الـ (GPU) في الحسابات هو اللاعب الرئيسي في تطور الأنظمة التي تعمل على التوازي بسبب طبيعتها التي خصصت في تنفيذ الحسابات الكثيفة والتي تعمل على التوازي. لبناء أنظمة ضخمة تعمل على التوازي ، هناك حاجة إلى استخدام مزيج من نماذج البرجمة المزدوجة و الثلاثية لزيادة مستوى التوازي في الأنظمة غير المتجانسة و التي تتضمن وحدات معاجلة مركزية (CPU) و وحدات معاجلة الرسوم (GPU) هناك عدد من نماذج البرمجة المزدوجة و المستخدمة في بناء الأنظمة و أحد هذه النماذج هو(MPI+OpenACC) هذا المزيج يحتوي على العديد من المزايا التي تساعد على زيادة مستوى التوازي في الأنظمة و كذلك يدعم العمل على مختلف المنصات و يساعد المبرمجين لبناء أنظمة متوازية بجهد أقل وعدد قليل من الأسطر البرمجية. أيضاً نموذج البرمجة (OpenACC) أصبح يستخدم مؤخراً بشكل متزايد في العديد من أجهزة الحاسبات العمالقة في جميع أنحاء العالم.

ومع ذلك، فإن بناء أنظمة ذات نماذج برمجة مختلفة مهمة صعبة، وتكون عرضة للأخطاء، ويصعب اختبارها. بالإضافة إلى ذلك فإن بناء أنظمة موازية باستخدام نموذج برمجة عالي المستوى يزيد من إمكانية حصول الأخطاء، وبالتالي فإن التطبيقات التي تعمل على التوازي لها سلوك غير محدد، مما يجعل اختبار واكتشاف الأخطاء أثناء وقت التشغيل مهمة صعبة. حتى بعد اكتشاف الأخطاء وتعديل شفرة المصدر، ليس من السهل تحديد ما إذا كان قد تم تصحيح الأخطاء أو إخفاؤها. إن دمج نموذجي مختلفين للبرجمة داخل نفس التطبيق يجعل من الصعب اختباره الا أن هذا التكامل قد يأتي بنوع جديد من الأخطاء أو سلوك مختلف. على الرغم من وجود العديد من أدوات الاختبار التي تكشف أخطاء وقت التشغيل، إلا أن هذا لا يزال غير مناسب لكشف الأخطاء التي تحدث في التطبيقات التي يتم تنفيذها في نماذج البرمجة المزدوجة وكذلك في نماذج البرمجة المتوازية عالية المستوى، وخاصة التطبيقات ذات الصلة بـ (OpenACC) نتيجة لذلك، يجب تحديد أخطاء الـ(OpenACC)التي لا يمكن الكشف عنها بواسطة الـ (Compilers)، ويجب توضيح أسبابها

في هذه الرسالة، نقوم باكتشاف وتحديد وتصنيف أخطاء وقت تشغيل لـ (OpenACC) وتحديد أسبابها مع شرح موجز لأول مرة يكون منشوراً في مجتمع البحث. كما نقدم ونقترح طرق اختبار جديدة للكشف عن أخطاء وقت التشغيل في الأنظمة المطبقة في نماذج البرمجة المزدوجة (MPI+OpenACC) وباستخدام لغة البرمجة (C++) ونطبق هذه التقنيات في أداة الاختبار الهجينة الموازية. تجمع التقنيات الهجينة بين تقنيات الاختبار الثابتة والديناميكية لاكتشاف أخطاء وقت التشغيل الحقيقية والمحتملة من خلال تحليل شفرة المصدر وأثناء وقت التشغيل. سيؤدي استخدام التقنيات الهجينة المتوازية إلى تعزيز وقت الاختبار وتغطية مجموعة واسعة من الأخطاء. أخيرا، على حد علمنا، لا يوجد أي عمل منشور حتى الآن يحدد أو يصنف الأخطاء المتعلقة بـ (OpenACC)، و لا توجد أداة اختبار مصممة لاختبار التطبيقات البرمجة باستخدام نموذج برمجة (OpenACC) أو نماذج البرمجة المزدوجة (MPI+OpenACC) واكتشاف أخطاء وقت التشغيل الخاصة بهم

Towards Exascale Software Testing

By

Ahmed Mohammed Alghamdi

Supervised by

Prof. Fathy Elbouraey Eassa

ABSTRACT

High-Performance Computing (HPC) recently has become important in several sectors, including the scientific and manufacturing fields. The continuous growth in building more powerful super-machines has become noticeable, and the Exascale supercomputer will be feasible in the next few years. As a result, building massively parallel systems becomes even more important to keep up with upcoming Exascale related technologies. Also, using GPU computation becomes the leading player in the evolution of parallel systems because of the nature of GPUs specialized for compute intensive and highly parallel computation. For building massively-parallel systems, a combination of programming models is needed to increase the system's parallelism, especially dual and tri-level programming models, to increase parallelism in heterogeneous systems that include CPUs and GPUs. There are several combinations of the dual-programming model; one of them is MPI+ OpenACC. This combination has several features that increase the application's parallelism concerning heterogeneous architecture and support different platforms with more performance, productivity, and programmability. In addition, OpenACC is a high-level parallel programming model used with FORTRAN, C, and C++ programming languages to accelerate the programmers' code with fewer changes and less effort, which reduces programmer workloads and makes it easier to use and learn. Also, OpenACC has been increasingly used in many top supercomputers around the world, and three of the top five HPC applications in Intersect360 Research are currently using OpenACC.

However, building systems with different programming models is a difficult task, error-prone and hard to test. In addition, building parallel systems by using a higher level programming model increases the possibility of introducing errors, and the parallel applications thus shows non-determined behavior, which makes testing and detecting runtime errors a challenging task. Even after detecting the errors and modifying the source code, it is not easy to determine whether the errors have been corrected or hidden. Integrating two different programming models inside the same application make it even more difficult to test because this integration could come with a new type of error or different behavior. Although there are many testing tools that detect runtime errors, this is still inadequate for detecting errors that occur in applications implemented in dual-programming models and in high-level parallel programming models,

especially OpenACC related applications. As a result, OpenACC errors that cannot be detected by compilers should be identified, and their causes should be explained.

In this thesis, we detect, identify, and classify OpenACC runtime errors and determine their causes with a brief explanation for the first time. Also, we provide and propose new testing techniques for detecting runtime errors in systems implemented in C++ and MPI + OpenACC dual-programming models, and we implement these techniques in our parallel hybrid-testing tool. The hybrid techniques combine static and dynamic testing techniques for detecting real and potential runtime errors by analyzing the source code and during runtime. Using parallel hybrid techniques will enhance the testing time and cover a wide range of errors. Finally, to the best of our knowledge, there is no published work to date that identifies or classifies OpenACC-related errors, nor is there a testing tool designed to test applications programmed by using the OpenACC programming model or the dual-programming models MPI + OpenACC and detect their runtime errors.