ADT K'Star

Action d'envergure C2S@Exa

Olivier Aumage
EPI RUNTIME, Inria Bordeaux – Sud-Ouest

Thierry Gautier
EPI MOAIS, Inria Rhône-Alpes

Sophia
12/2013

Caractéristiques

ADT K'Star

- ADT sur le contingent national
- Participants
  - EPI MOAIS
    - Thierry Gautier (porteur de l'action)
    - Pierrick Brunet (ADT K'Star), Philippe Virouleau (ADT KAWAH)
    - François Broquedis, Vincent Danjean
  - EPI RUNTIME
    - Olivier Aumage (co-responsable)
    - Samuel Thibault, Nathalie Furmento
- Durée
  - 2 ans
- Moyens
  - 1 ingénieur (IJD) sur 2 ans à Montbonnot
    - Pierrick Brunet, ENST Bretagne, prise de fonctions: mi-octobre 2013
  - 1 IJD sur 1 an à Bordeaux, à recruter sur la campagne 2014
    - Cadre Inria Bordeaux « HPC Collectif »

Introduction

Supports exécutifs pour le calcul intensif

- Ordonnancement de tâches
  - plateformes multicœurs
  - supports d'accélérateurs
- StarPU
  - EPI RUNTIME, Bordeaux
- XKaapi
  - EPI MOAIS, Montbonnot

Objectifs

Compilation source à source

- Faciliter l'adaptation de logiciels au-dessus des supports exécutifs
  - Courbe d'apprentissage
  - Portabilité
- Traduire des directives de type « pragma »
  - C, C++, Fortran
  - Annotations
- Vers des appels à StarPU ou XKaapi
- En s'appuyant sur un standard
  - OpenMP 3.1 + fonctionnalités 4.0 (tâches dépendantes, accélérateurs)
    - C, C++, Fortran
  - Possibilité d'extensions
    - Mise en œuvre d'optimisations
    - Démonstrateur

Contexte

Mise en œuvre de compilateurs

- Travaux existants dans les EPI
  - EPI MOAIS
    - Utilisation de l'architecture CLang pour piloter XKaapi
      ADT KAWAH (2012-2014)
    - Anciennement, exploration de l'architecture de compilation ROSE
  - EPI RUNTIME
    - Plugin GCC de programmation de StarPU
    - Ludovic Courtès
      ADT Programmation des architectures hétérogènes (2010-2012)

Exemple sur l'existant

Plugin GCC de StarPU

```
/* Codelet */
extern void my_task (char, const char *, float *, int)        __attribute__ ((task));

/* CPU version declaration */
static void my_task_cpu (char, const char *, float *, int)    __attribute__ ((task_implementation
                                                                    ("cpu", my_task)));

/* OpenCL version declaration
static void my_task_opencl (char, const char *, float *, int) __attribute__ ((task_implementation
                                                                     ("opencl", my_task)));

   . . .

int main(int argc, char *argv[]) {
        /* Data declaration */
        char  in [argc * 3][42]  __attribute__ ((heap_allocated));
        float out[argc * 3][42]  __attribute__ ((heap_allocated));

        /* Task instanciation */
        my_task (42, in, out, argc * 3);
        #pragma starpu wait

        return 0;
}
```

État des lieux

Langages

- Langages à directives
  - OpenMP
  - OpenACC
  - XcalableMP
  - etc.

- Langages spécifiques
  - Nvidia Cuda
  - OpenCL
  - Cilk, Intel Cilk+
  - etc.

- Considérations
  - Être au plus proche des codes applicatifs existant

État des lieux

Compilateurs

- GNU GCC
  - DragonEgg
- LLVM / CLang
- Rose
- Considérations
  - Compiler en source-to-source
    - Pouvoir bénéficier de compilateurs natifs existants

Exemple de code

Hello World…

```
#include <stdio.h>

void hello_world()
{
    printf("Hello !\n");
}

int main( int argc, char** argv)
{
#pragma omp task
        hello_world();

#pragma omp taskwait
        printf("Done\n");

    return 0;
}
```

Exemple de code

Cholesky

```
#include <cblas.h>
#include <clapack.h>
void Cholesky( int N, double A[N][N], size_t NB ) {
#pragma omp parallel
#pragma omp single
  for (size_t k=0; k < N; k += NB)
  {

    clapack_dpotrf( CblasRowMajor, CblasLower, NB, &A[k][k], N );

    for (size_t m=k+ NB; m < N; m += NB)
    {


      cblas_dtrsm ( CblasRowMajor, CblasLeft, CblasLower, CblasNoTrans, CblasUnit,
        NB, NB, 1., &A[k*N+k], N, &A[m*N+k], N );
    }

    for (size_t m=k+ NB; m < N; m += NB)
    {


     cblas_dsyrk ( CblasRowMajor, CblasLower, CblasNoTrans,
        NB, NB, -1.0, &A[m*N+k], N, 1.0, &A[m*N+m], N );

      for (size_t n=k+NB; n < m; n += NB)
      {


        cblas_dgemm ( CblasRowMajor, CblasNoTrans, CblasTrans,
          NB, NB, NB, -1.0, &A[m*N+k], N, &A[n*N+k], N, 1.0, &A[m*N+n], N );
      }
    }
  }
}
```

Exemple de code

Cholesky

```
#include <cblas.h>
#include <clapack.h>
void Cholesky( int N, double A[N][N], size_t NB ) {
#pragma omp parallel
#pragma omp single
  for (size_t k=0; k < N; k += NB)
  {
#pragma omp task depend(inout: &A[k:NB][k:NB])
    clapack_dpotrf( CblasRowMajor, CblasLower, NB, &A[k][k], N );

    for (size_t m=k+ NB; m < N; m += NB)
    {
#pragma omp task depend(in:&A[k:NB][k:NB]) \
                 depend(inout:&A[m:NB][k:NB])
      cblas_dtrsm ( CblasRowMajor, CblasLeft, CblasLower, CblasNoTrans, CblasUnit,
        NB, NB, 1., &A[k*N+k], N, &A[m*N+k], N );
    }

    for (size_t m=k+ NB; m < N; m += NB)
    {
#pragma omp task depend(in:&A[m:NB][k:NB]) \
                 depend(inout:&A[m:NB][m:NB])
     cblas_dsyrk ( CblasRowMajor, CblasLower, CblasNoTrans,
        NB, NB, -1.0, &A[m*N+k], N, 1.0, &A[m*N+m], N );

      for (size_t n=k+NB; n < m; n += NB)
      {
#pragma omp task depend(in:&A[m:NB][k:NB], &A[n:NB][k:NB])\
                 depend(inout:&A[m:NB][n:NB])
        cblas_dgemm ( CblasRowMajor, CblasNoTrans, CblasTrans,
          NB, NB, NB, -1.0, &A[m*N+k], N, &A[n*N+k], N, 1.0, &A[m*N+n], N );
      }
    }
  }
}
```

Exemple de code

omp target / omp target data

```
subroutine vec_mult(p, v1, v2, N)
   real    :: p(N), v1(N), v2(N)
   integer :: i

   call init(v1, v2, N)

   !$omp target data map(from: p)

      !$omp target map(to: v1, v2 )
         !$omp parallel do
         do i=1,N
           p(i) = v1(i) * v2(i)
         end do
      !$omp end target

      call init_again(v1, v2, N)

      !$omp target map(to: v1, v2)
         !$omp parallel do
         do i=1,N
            p(i) = p(i) + v1(i) * v2(i)
         end do
      !$omp end target

   !$omp end target data

   call output(p, N)
end subroutine
```

Traduction

OpenMP → Xkaapi / StarPU

- task, taskwait
  - création de tâches XKaapi ou StarPU, synchronisation
- régions parallèles, barrières
  - idem
- worksharing (for, single, master, section)
  - supports spécifiques dans les runtimes

Planning

Calendrier prévisionnel

- Extension de CLANG suffisantes : Q1 2014
  - Intel : support du parsing OpenMP-3.1
  - Ajout d'un support adapté pour le multi-versionning des tâches
    - Norme OpenMP-4.0 trop directive ne laissant pas le choix d'ordonnancement par le runtime ?

- Q2 2014 :
  - Validation et optimisation
  - Expérimentation sur des tests et micro-benchmarks C/C++
  - Diffusion V0

- Q3-Q4 2014 :
  - Expérimentation sur des applications C/C++
  - Diffusion V1

- Q4 2014 - 2015 :
  - Fortran important, mais 1 ingénieur sur les 2 prévus
  - Demande spécifique ?

À retenir

ADT K'STAR conjointe MOAIS (Pole 4) – RUNTIME (Pole 3)

a) Objectifs de l'ADT

- Compiler OpenMP 4.0 en source-à-source vers Starpu ou XKaapi
- Faciliter l'adaptation portable des codes au-dessus des supports exécutifs
- Augmenter la visibilité et la diffusion de StarPU et XKaapi

b) Logiciels visés

- Codes applicatifs et solveurs C2S@Exa en première approche

c) Personnes impliquées (chercheurs et ingénieurs)

- Portée par Thierry Gautier, EPI MOAIS
- Inria Grenoble (Thierry Gautier, François Broquedis, Vincent Danjean) + Inria Bordeaux (Olivier Aumage, Samuel Thibault, Nathalie Furmento)
- Pierrick Brunet, ingénieur (2 ans) à Montbonnot
- + 1 ingénieur IJD (1an) à recruter à Bordeaux lors de la campagne 2014

d) Avancement des travaux

- …/…

À retenir

ADT K'STAR conjointe MOAIS (Pole 4) – RUNTIME (Pole 3)

d) Avancement des travaux

- T0 + 1.5 mois, ~ 3000 lignes de code
- Extension dans CLANG pour le traitement des dépendances
  - sur des variables
  - sur des sous-tableaux <= important !
- Factorisation du code de génération + spécialisation pour XKaapi ou StarPU
- Projet sur la Forge INRIA
  - tests unitaires
  - test simple en algèbre linéaire (Cholesky)
  - dépôt GIT
- Version fonctionnelle mi-décembre
  - tâches, synchronisation, région parallèle, single & master
  - limitations : threadprivate (sur les data), boucles (omp for)