StarPU

Edênis Freindorfer Azevedo

Instituto de Matemática e Estatística

10 de junho de 2015

Principais ideias

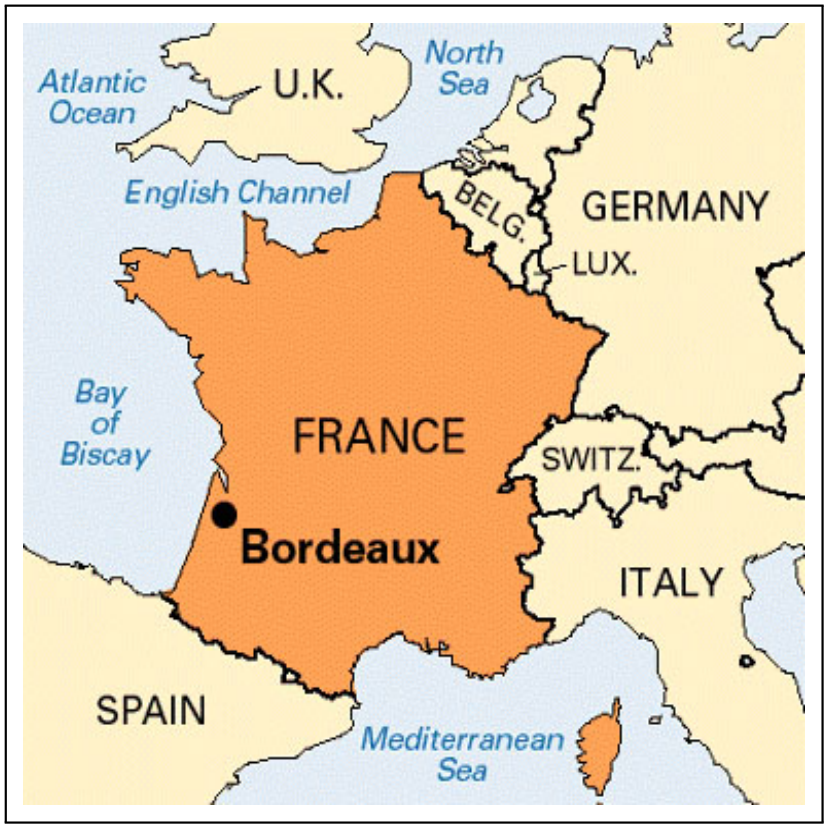
1. Entender o que é *StarPU*.
2. Entender como *StarPU* funciona.

Resumo

- *StarPU* é uma biblioteca de programação para arquiteturas híbridas.
- Possui um sistema de escalonamento embutido.

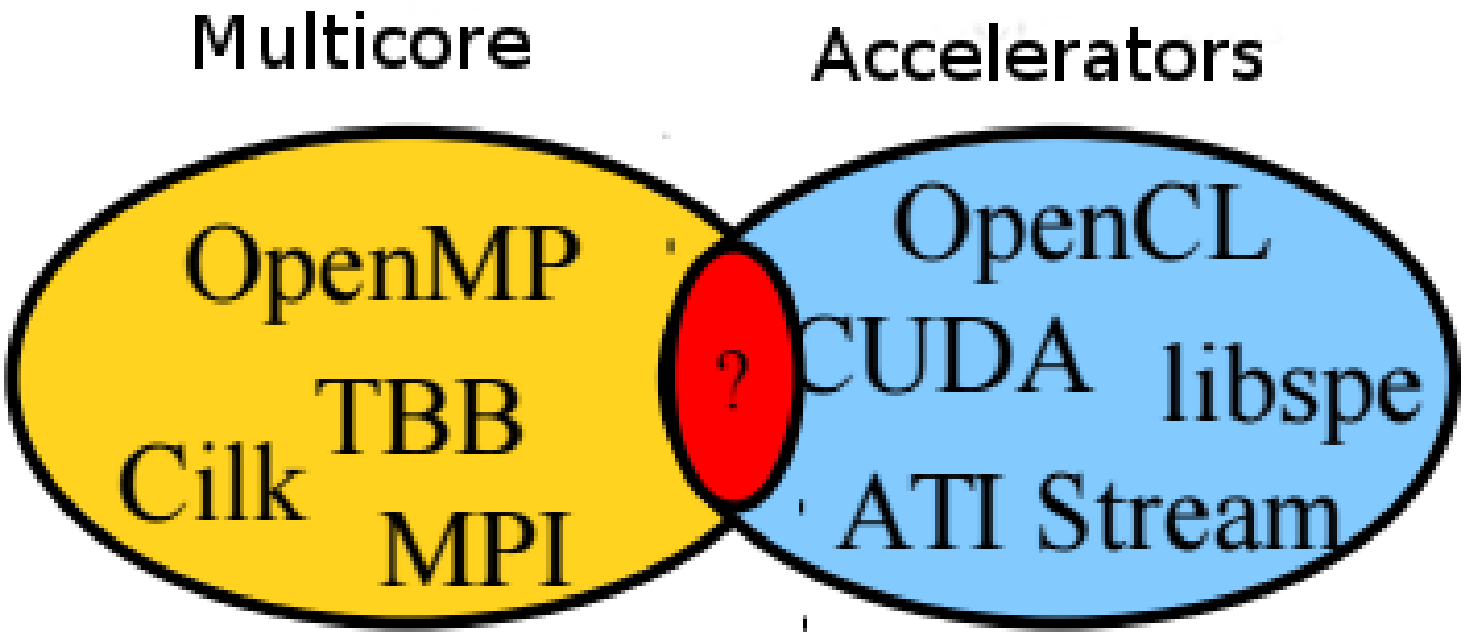
Origem

- Desenvolvido pelo **INRIA**.
- Iniciado em 2009.
- Versão 1.1.4 lançada em março de 2015.
- Open Source – Licença *LGPL*



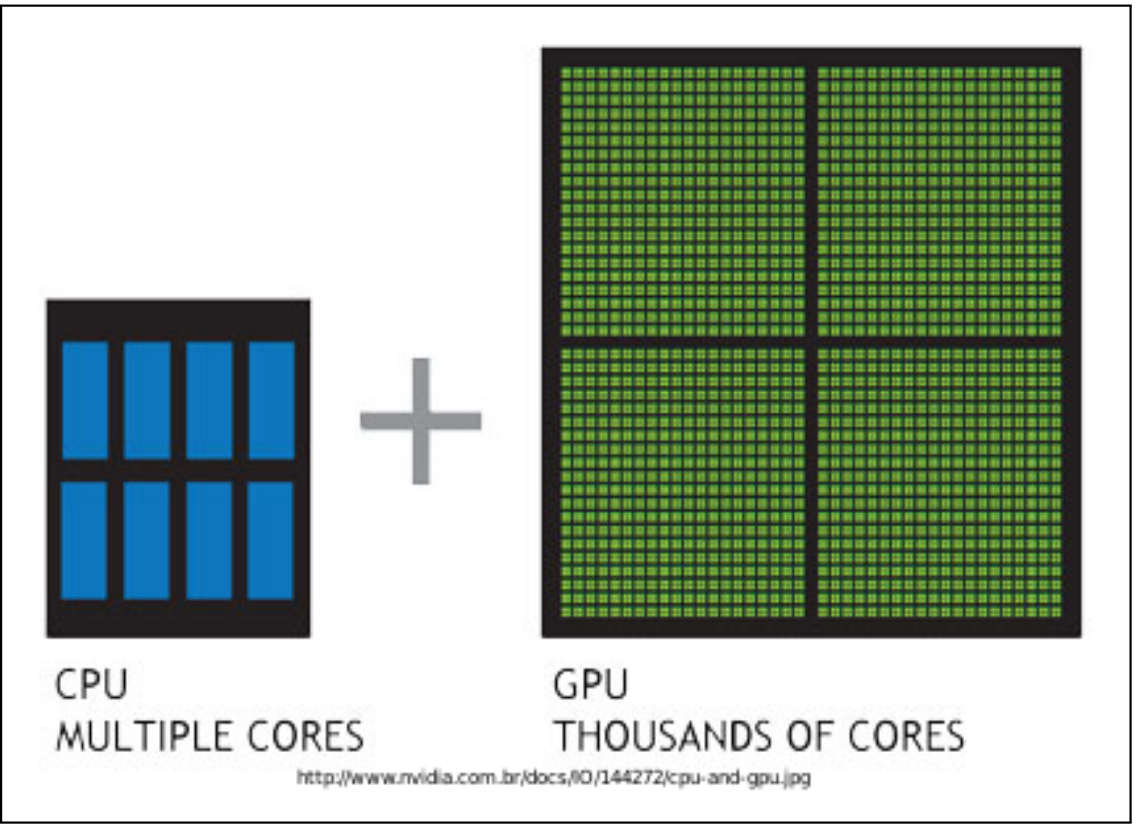
Contexto

Dadas duas matrizes, como fazer a multiplicação/soma/subtração?



http://www.x.org/wiki/Events/XDC2014/XDC2014ThibaultStarPU/xdc_starpu.pdf

Motivo I


CPU
MULTIPLE CORES
GPU
THOUSANDS OF CORES
http://www.nvidia.com.br/docs/IO/144272/cpu-and-gpu.jpg


TOP500
SUPERCOMPUTER SITES

Motivo II

"While a lot of efforts have been devoted to offload computation onto such accelerators, very little attention has been paid to portability concerns on the one hand, and to the possibility of having heterogeneous accelerators and processors to interact on the other hand."

Fonte: http://starpu.gforge.inria.fr/doc/html/index.html

Objetivos do StarPU

1. Visão unificada dos recursos computacionais
2. Escalonamento eficiente e/ou personalizado

Implementação

1. Existe um plugin para o compilador *GCC*
2. Biblioteca apenas para a linguagem *C*

Hello World – plugin GCC

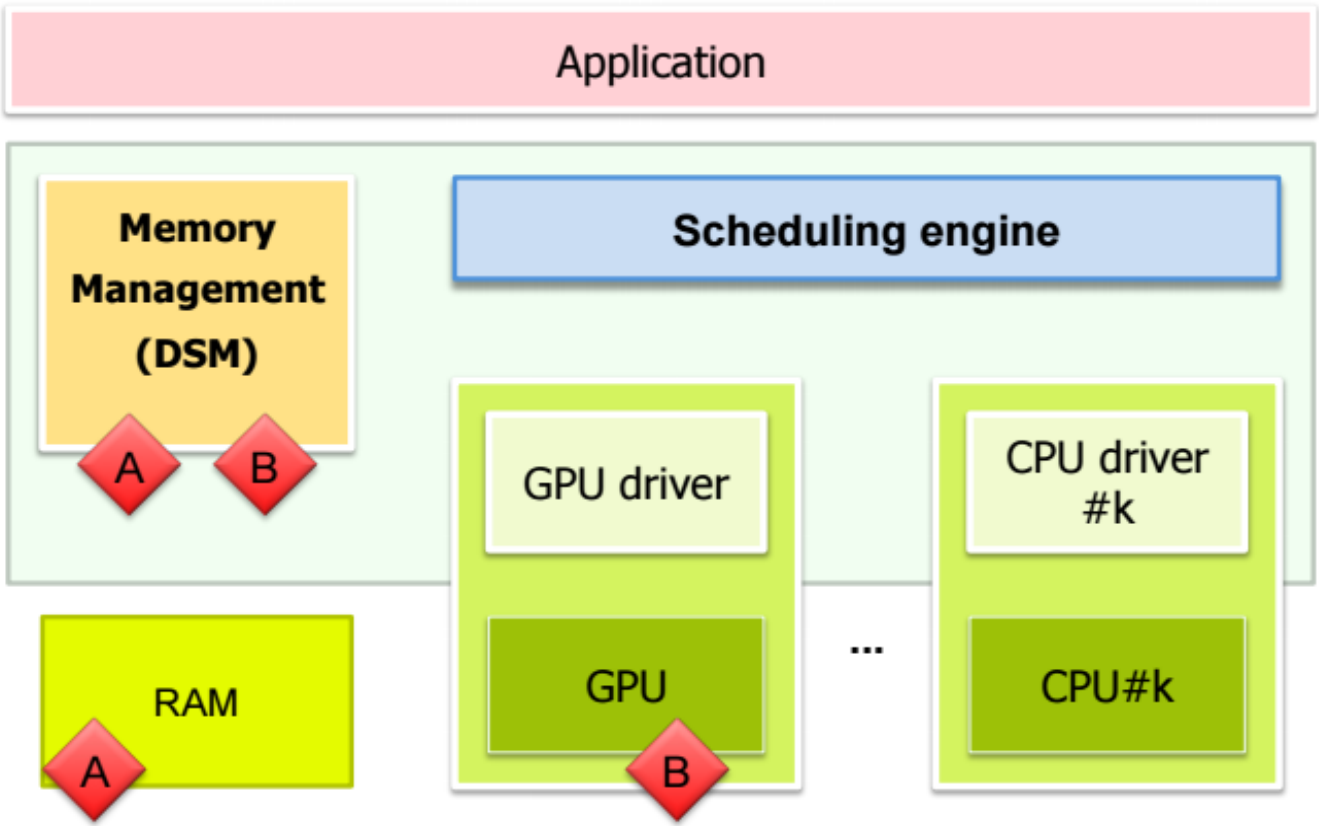
```
#include <stdio.h>

/* Task declaration.  */
static void my_task (int x) __attribute__ ((task));

/* Definition of the CPU implementation of `my_task'.  */
static void my_task (int x)
{
  printf ("Hello, world!  With x = %d\n", x);
}

int main ()
{
  /* Initialize StarPU. */
#pragma starpu initialize

  /* Do an asynchronous call to `my_task'. */
  my_task (42);

  /* Wait for the call to complete.  */
#pragma starpu wait

  /* Terminate. */
#pragma starpu shutdown

  return 0;
}
```

Estruturas chaves

1. **Codelet**: kernel implementado em múltiplas arquiteturas
2. **Task**: aplicação de um Codelet em um conjunto de dados
3. **Memória Virtual de Software**: disponibiliza o acesso aos dados de maneira uniforme

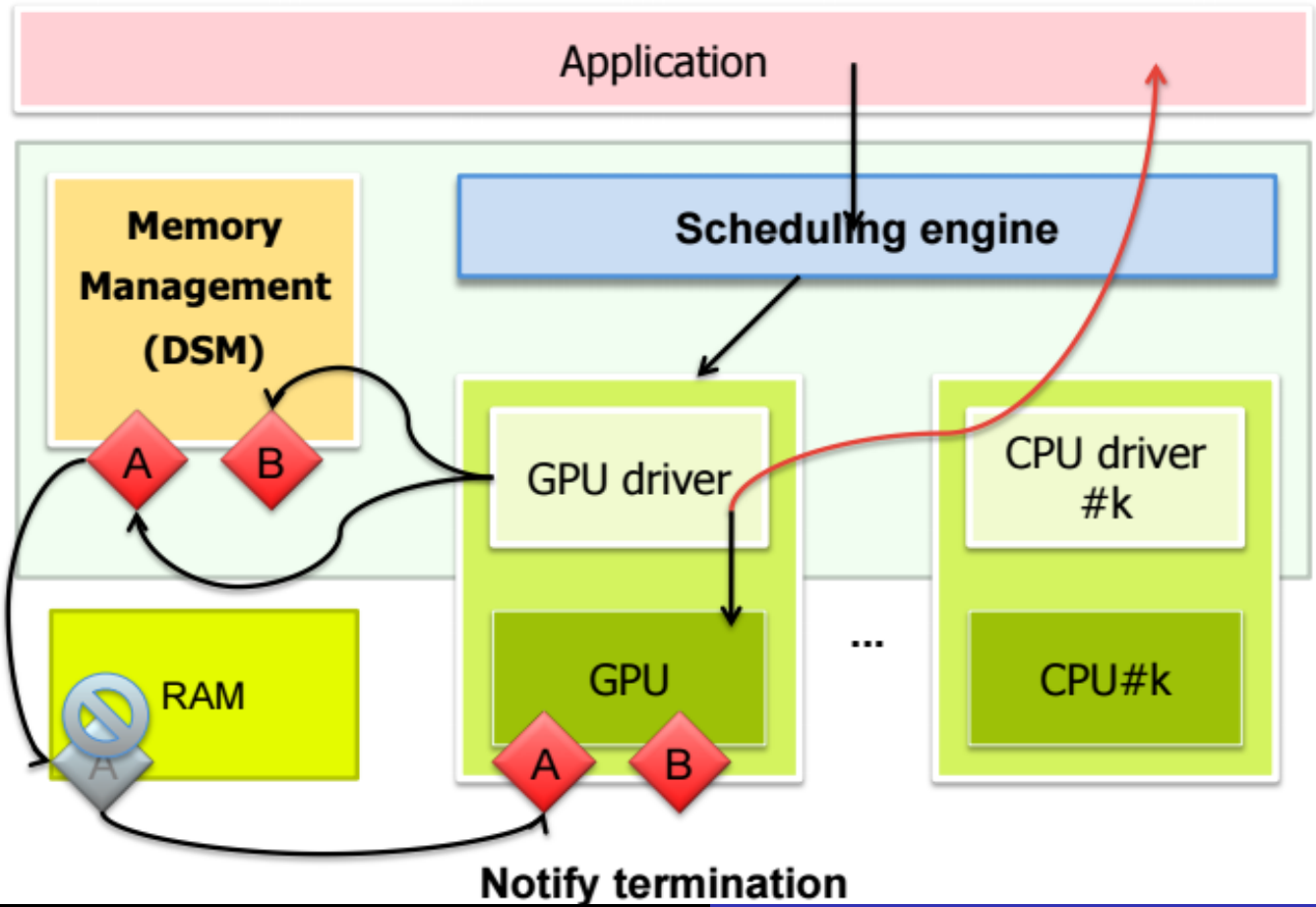
Panorama

http://starpu.gforge.inria.fr/tutorials/2015-01-HiPEAC/hipeac_tutorial_hetcomp_starpu_2015.pdf



Panorama

http://starpu.gforge.inria.fr/tutorials/2015-01-HiPEAC/hipeac_tutorial_hetcomp_starpu_2015.pdf



Codelet

```
struct starpu_codelet cl =
{
        .cpu_funcs = { OP_cpu_func, OP_sse_func },
        .cuda_funcs = { OP_cuda_func },
        .opencl_funcs = { OP_opencl_func },
        .nbuffers = 1,
        .modes = { STARPU_RW }
};
```

Aplicação

Multiplicação de um vetor por um escalar

- Facilmente paralelizável!
- Exemplo de todos os tutoriais

Codelet

```
struct starpu_codelet cl =
{
        .cpu_funcs = { scal_cpu_func, scal_sse_func },
        .cuda_funcs = { scal_cuda_func },
        .scalencl_funcs = { scal_scalencl_func },
        .nbuffers = 1,
        .modes = { STARPU_RW }
};
```

Disponibilizando na estrutura de memória

```
float vector[NX];
/* ... fill data ... */

starpu_data_handle_t vector_handle;
starpu_vector_data_register(&vector_handle, 0,
                            (uintptr_t)vector, NX, sizeof(vector[0]));

/* ... use the vector through the handle ... */

starpu_data_unregister(vector_handle);
```

Kernel

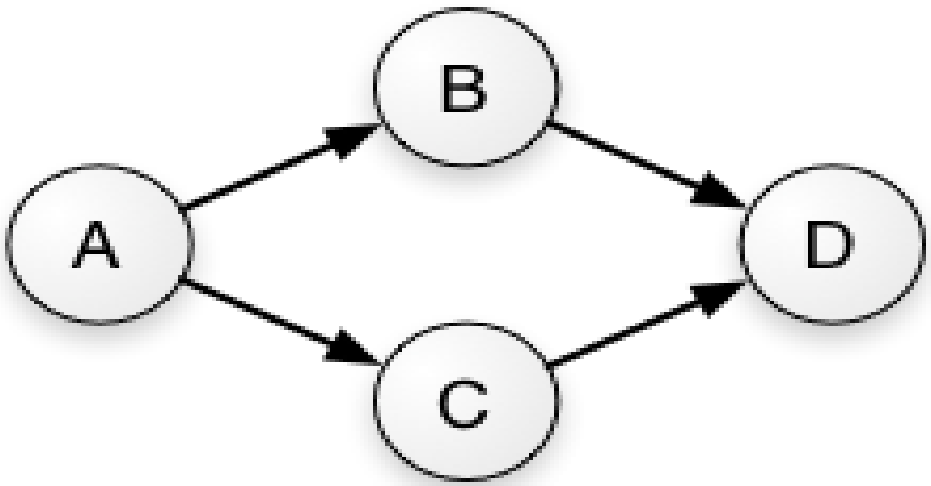
```
void scal_cpu_func(void *buffers[], void *cl_arg) {
    struct starpu_vector_interface *vector_handle = buffers[0];

    unsigned n     = STARPU_VECTOR_GET_NX(vector_handle);
    float *vector = STARPU_VECTOR_GET_PTR(vector_handle);

    float *ptr_factor = cl_arg;

    unsigned i;
    for (i = 0; i < n; i++)
        vector[i] *= *ptr_factor;
}
```

Execução

```
starpu_task_insert(
                &scal_cl,
                STARPU_RW, vector_handle,
                STARPU_VALUE, &factor, sizeof(factor),
                0);

starpu_task_wait_for_all();
```

Precedência de tarefas

Utilizando grafo de precedências de "tasks"



http://www.ppgia.pucpr.br/ laplima/ensino/soeen/so_common/materia/images/ps-grafo_precedencia.png

Funções prontas

- Fatoração de matrizes (*Cholesky/QR/LU, FFT*)
- Processadores especiais (*Xeon Phi*)
- Acesso ao disco rígido (*starpu_disk_ops*)

Escalonamento

- Personalizável!
- Passado como parâmetro:
  $ STARPU_SCHED=greedy ./foo

MPI

- Substituição de "*void **" por "*starpu_data_handle_t*" nos protótipos das funções.
- Sem função de *broadcast*.

MPI	STARPU
MPI_Scatter	starpu_mpi_scatter_detached
MPI_Gather	starpu_mpi_gather_detached
MPI_Send	starpu_mpi_send
MPI_Recv	starpu_mpi_recv

Geral

- É uma camada de abstração!
- O programador pode se concentrar só no algoritmo!

É utilizável?

SIM!

- Fácil de instalar (apt-get install libstarpu-dev)
- Compatível com Linux, Windows e Mac OS
- Muito bem documentado
  `http://starpu.gforge.inria.fr/doc/html/index.html`

Quem usa?

- MAGMA (álgebra linear)
- Chameleon (álgebra linear)
- PasTiX (álgebra linear)
- SkePU (biblioteca de programação para GPU)

Referências Bibliográficas

- The StarPU Runtime System. Accessado em 8.6.15. Disponível em: <http://starpu.gforge.inria.fr/tutorials/2015-01-HiPEAC/hipeac_tutorial_hetcomp_starpu_2015.pdf>
- Augonnet et al. **StarPU: Seamless Computations Among CPUs and GPUs**. Accessado em 8.6.15. Disponível em: <http://www.x.org/wiki/Events/XDC2014/XDC2014ThibaultStarPU/xdc_starpu.pdf>
- StarPU Handbook Accessado em 8.6.15. Disponível em: <http://starpu.gforge.inria.fr/doc/html/index.html>

Dúvidas?

denisfa@ime.usp.br