Computational Experiments on Task-Based Parallel Applications

Salão de Iniciação Científica UFRGS 2019

Henrique Corrêa Pereira da Silva
Lucas Mello Schnorr (advisor)

.inf INSTITUTO DE INFORMÁTICA UFRGS

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL UFRGS

Context

In the last few years, we saw the continuation of a decades long trend: ever-growing parallelism in search of performance

Context

In the last few years, we saw the continuation of a decades long trend: ever-growing parallelism in search of performance

Also we saw that. . .

Context

In the last few years, we saw the continuation of a decades long trend: ever-growing parallelism in search of performance

Also we saw that. . .

- the hardware naturally specialized over time

Context

In the last few years, we saw the continuation of a decades long trend: ever-growing parallelism in search of performance

Also we saw that. . .

- the hardware naturally specialized over time
- buying highly performant systems became very expensive

Context

In the last few years, we saw the continuation of a decades long trend: ever-growing parallelism in search of performance

Also we saw that...

- the hardware naturally specialized over time
- buying highly performant systems became very expensive
- more common systems configurations became hard to extract their full potential

Context

In the last few years, we saw the continuation of a decades long trend: ever-growing parallelism in search of performance

Also we saw that...

- the hardware naturally specialized over time
- buying highly performant systems became very expensive
- more common systems configurations became hard to extract their full potential

Applications running on them can no longer rely on homogeneous hardware if they seek *high performance*

Reality

Thus, a solution is needed in face of this demand. . .

Reality

Thus, a solution is needed in face of this demand. . .

Some sort of middle layer capable of partitioning applications workloads into these hybrid systems!

Agenda

Agenda

Most popular

There are several popular parallel programming models

Most popular

There are several popular parallel programming models

Examples:

Most popular

There are several popular parallel programming models

Examples:

- Message passing

Most popular

There are several popular parallel programming models

Examples:

- Message passing
- Shared memory

Most popular

There are several popular parallel programming models

Examples:

- Message passing
- Shared memory
- Hybrid model

Most popular

There are several popular parallel programming models

Examples:

- Message passing
- Shared memory
- Hybrid model

In any of those models, their implementation is a manual, complex, time-consuming and, therefore, error-prone process

Implementations

Most popular implementations seek to alleviate the burden of programmers

Implementations

Most popular implementations seek to alleviate the burden of programmers

Of those implementations the most popular are:

Implementations

Most popular implementations seek to alleviate the burden of programmers

Of those implementations the most popular are:

- OpenMP
- OpenMPI
- OpenACC
- etc..

Implementations

Most popular implementations seek to alleviate the burden of programmers

Of those implementations the most popular are:

- OpenMP
- OpenMPI
- OpenACC
- etc..

Attention!

If only utilizing the previously cited APIs, the domain decomposition is normally fixed to the number of resources

Implementations

Most popular implementations seek to alleviate the burden of programmers

Of those implementations the most popular are:

- OpenMP
- OpenMPI
- OpenACC
- etc..

Attention!

If only utilizing the previously cited APIs, the domain decomposition is normally fixed to the number of resources you'll be victim to dynamic load imbalances

StarPU

One of the efforts into creating a middleware capable of overcoming said limitations is called StarPU

StarPU

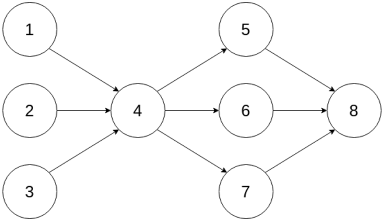
One of the efforts into creating a middleware capable of overcoming said limitations is called StarPU

Its approach is by defining the problem by tasks, and those tasks into a *Directed Acyclic Graph* (or *DAG* for short)

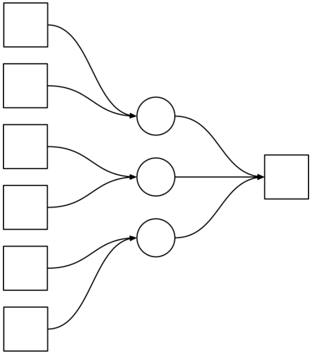
StarPU

One of the efforts into creating a middleware capable of overcoming said limitations is called StarPU

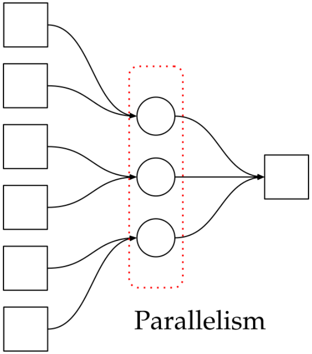
Its approach is by defining the problem by tasks, and those tasks into a *Directed Acyclic Graph* (or *DAG* for short)



Task-based



Task-based



Agenda

Intent

Our objectives were to

Intent

Our objectives were to

1. Use the task-based approach

Intent

Our objectives were to

1. Use the task-based approach
2. Learn the StarPU API

Intent

Our objectives were to

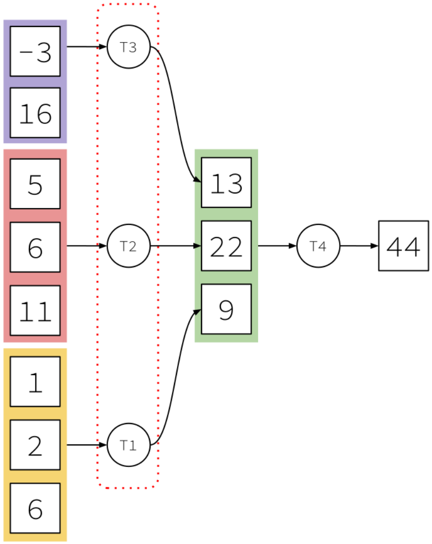
1. Use the task-based approach
2. Learn the StarPU API
3. Analyze its behavior while executing the application

Object of study

A simple vector accumulation

Object of study

A simple vector accumulation



Example kernel

```c
void reduc_sum(void** buffers, void* cl_arg)
{
    ullint* vec_input = (ullint*)STARPU_VECTOR_GET_PTR(buffers[0]);
    ullint* output = (ullint*)STARPU_VARIABLE_GET_PTR(buffers[1]);
    uint nx_input = STARPU_VECTOR_GET_NX(buffers[0]);

    double t0 = get_time();

    // do the job
    for (uint i = 0; i < nx_input; i++)
        *output += vec_input[i];

    double t1 = get_time();

    V_PRINTF("SUM = %d\n"
             "Task finished work with elapsed time %f\n",
        *output, t1 - t0);
}
```

Agenda

Basis of comparison

Other simple implementations of vector accumulations

Basis of comparison

Other simple implementations of vector accumulations

1. Naive
2. C++ STL
3. OpenMP

Basis of comparison

Other simple implementations of vector accumulations

1. Naive
2. C++ STL
3. OpenMP

Not intended as a goal to beat, but as a basis

Methodological approach

A full factorial, randomly ordered experiment design

Methodological approach

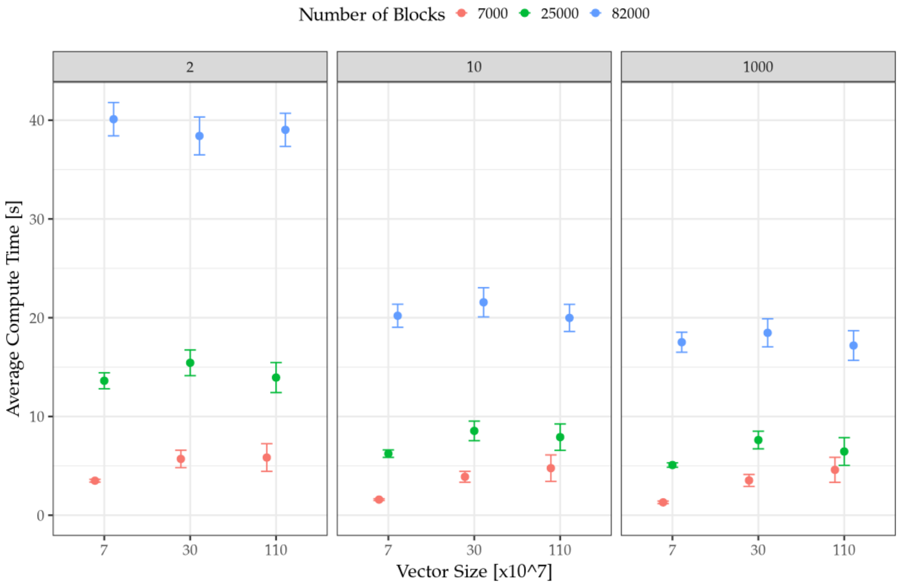
A full factorial, randomly ordered experiment design

Parameters:

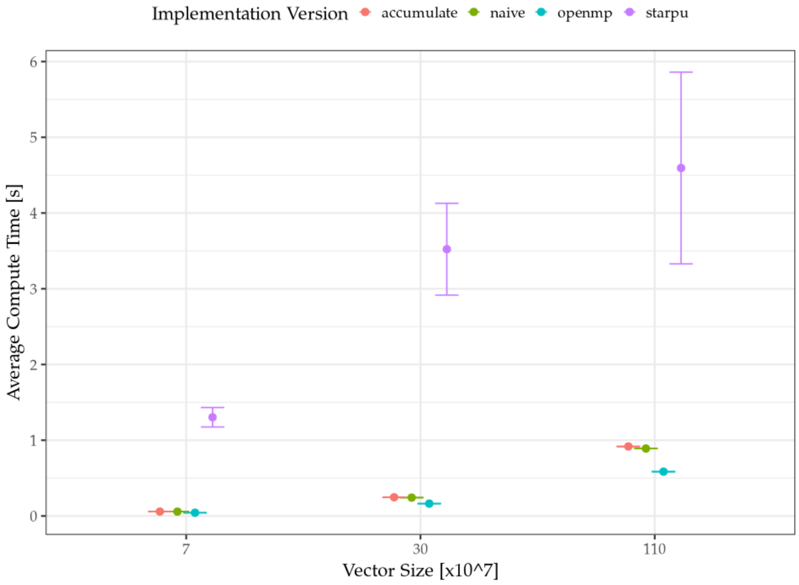
- *Vector size*: `7*10^7, 3*10^8 and 1.1*10^9`
- *Number of blocks*: `7000, 25000 and 82000`
- *Reduction factor*: `2, 10 and 1000`

Agenda

StarPU


Number of Blocks
7000
25000
82000
2
10
1000
Average Compute Time [s]
0
10
20
30
40
7
30
110
Vector Size [x10^7]

Combined graph



Agenda

Feasibility

Even with a simple implementation, we have shown that StarPU is a very capable API

Feasibility

Even with a simple implementation, we have shown that StarPU is a very capable API

Furthermore, we can aggregate the other APIs into our computation kernel

Feasibility

Even with a simple implementation, we have shown that StarPU is a very capable API

Furthermore, we can aggregate the other APIs into our computation kernel

OpenMP	therefore, utilizing *parallel tasks*
OpenMPI	distribute the execution graph across a whole cluster

Questions?

Salão de Iniciação Científica UFRGS 2019

Henrique Corrêa Pereira da Silva
Lucas Mello Schnorr (advisor)

.inf INSTITUTO DE INFORMÁTICA UFRGS

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL UFRGS