SOCL

OpenCL Frontend for StarPU

Sylvain HENRY
sylvain.henry@inria.fr

University of Bordeaux - LaBRI - Inria

April 5th, 2013

StarPU

How to use the StarPU runtime system:

1. Low-level API

```
starpu_data_handle_t vector_handle;

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

struct starpu_task *task = starpu_task_create();
task->synchronous = 1;
task->cl = &cl;
task->handles[0] = vector_handle;
task->cl_arg = &factor;
task->cl_arg_size = sizeof(factor);

starpu_task_submit(task);

starpu_data_unregister(vector_handle);
```

StarPU

How to use the StarPU runtime system:

1. Low-level API
2. Insert Task

```
starpu_insert_task(&mandelbrot_cl,
      STARPU_VALUE, &iby, sizeof(iby),
      STARPU_VALUE, &block_size, sizeof(block_size),
      STARPU_VALUE, &stepX, sizeof(stepX),
      STARPU_VALUE, &stepY, sizeof(stepY),
      STARPU_W, block_handles[iby],
      STARPU_VALUE, &pcnt, sizeof(int *),
      0);
```

StarPU

How to use the StarPU runtime system:

1. Low-level API
2. Insert Task
3. GCC Plugin

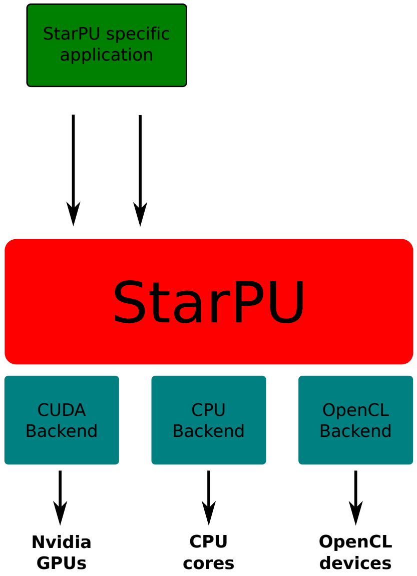
```
static void matmul (const float *A, const float *B, float *C, size_t nx, size_t
      ny, size_t nz)
  __attribute__ ((task));

static void matmul_cpu (const float *A, const float *B, float *C, size_t nx,
      size_t ny, size_t nz)
  __attribute__ ((task_implementation ("cpu", matmul)));

#pragma starpu register &A[i*zdim*bydim + j*bzdim*bydim] (bzdim * bydim)
#pragma starpu register &B[i*xdim*bzdim + j*bxdim*bzdim] (bxdim * bzdim)
#pragma starpu register &C[i*xdim*bydim + j*bxdim*bydim] (bxdim * bydim)

matmul (&A[i * zdim * bydim + k * bzdim * bydim],
      &B[k * xdim * bzdim + j * bxdim * bzdim],
      &C[i * xdim * bydim + j * bxdim * bydim], bxdim,
      bydim, bzdim);

#pragma starpu wait
```

StarPU

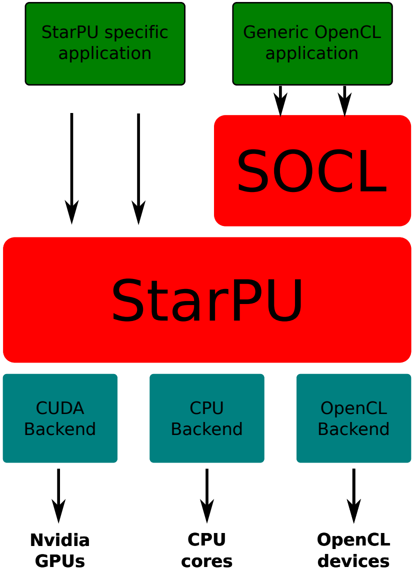
How to use the StarPU runtime system:

1. Low-level API
2. Insert Task
3. GCC Plugin
4. **OpenCL API**

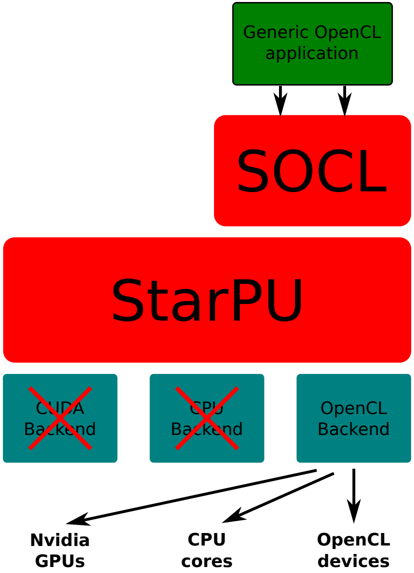
Layers


StarPU specific application
StarPU
CUDA Backend
CPU Backend
OpenCL Backend
Nvidia GPUs
CPU cores
OpenCL devices

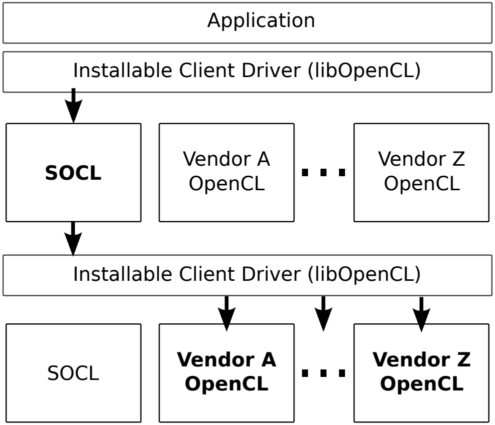
Layers


StarPU specific application
Generic OpenCL application
SOCL
StarPU
CUDA Backend
CPU Backend
OpenCL Backend
Nvidia GPUs
CPU cores
OpenCL devices

Layers


Generic OpenCL application
SOCL
StarPU
CUDA Backend
CPU Backend
OpenCL Backend
Nvidia GPUs
CPU cores
OpenCL devices

Installable Client Driver



SOCL Platform

- SOCL Platform exposes devices of every other platforms
    - GPUs, CPUs, accelerators. . .

SOCL Platform

- SOCL Platform exposes devices of every other platforms
  - GPUs, CPUs, accelerators. . .
- Synchronizations (events. . .) can be used between all devices

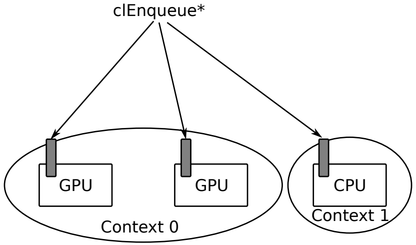
SOCL Platform

- SOCL Platform exposes devices of every other platforms
  - GPUs, CPUs, accelerators. . .
- Synchronizations (events. . .) can be used between all devices
- Buffers can be shared by all devices
  - Automatic transfers between devices handled by StarPU

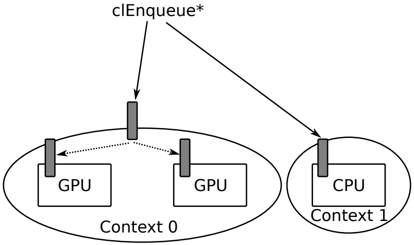
SOCL Platform

- SOCL Platform exposes devices of every other platforms
  - GPUs, CPUs, accelerators...
- Synchronizations (events...) can be used between all devices
- Buffers can be shared by all devices
  - Automatic transfers between devices handled by StarPU
- Static scheduling (à-la OpenCL) can be used as described in the specification
  - One command queue per device

Scheduling Contexts

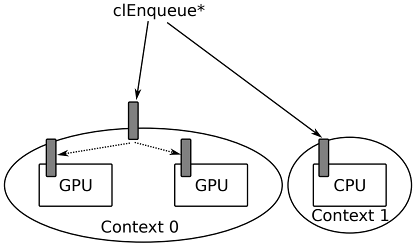


Scheduling Contexts



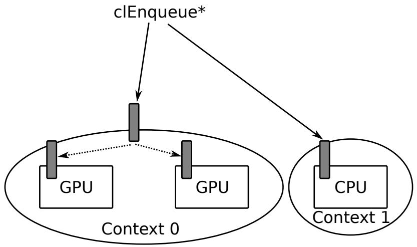
- Command queues can now be associated to contexts

Scheduling Contexts



- Command queues can now be associated to contexts
- Automatic scheduling for commands submitted in these queues!

Scheduling Contexts

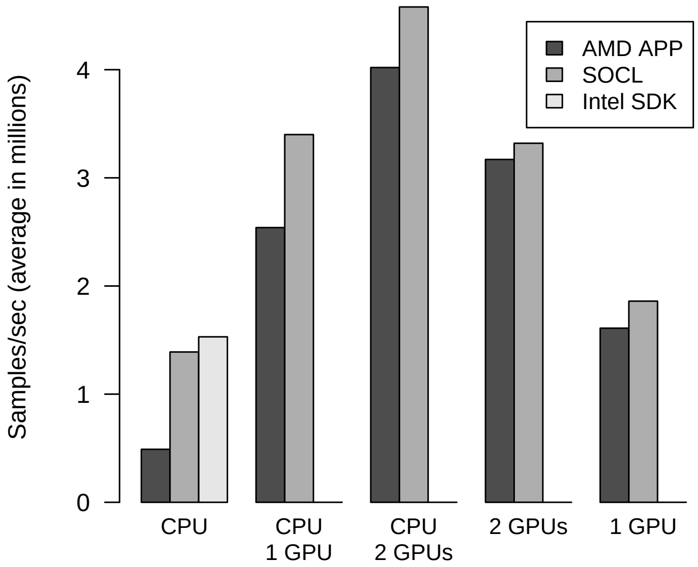


- Command queues can now be associated to contexts
- Automatic scheduling for commands submitted in these queues!
- Scheduler can be chosen for each context

Scheduling Contexts

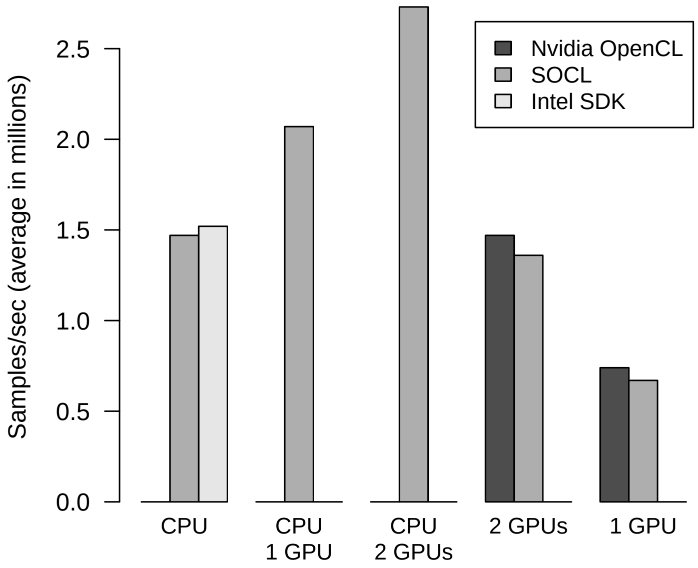
```
cl_context_properties gpu_properties[] = {
   CL_CONTEXT_PLATFORM, (cl_context_properties)platforms[platform_idx],
   CL_CONTEXT_SCHEDULER_SOCL, "heft",
   CL_CONTEXT_NAME_SOCL, "GPUs",
   0
};
gpu_context = clCreateContextFromType(gpu_properties, CL_DEVICE_TYPE_GPU, NULL,
     NULL, &err);
```

LuxRender



Intel Xeon E5-2650 2.00GHz with 32GB, 2 AMD Radeon HD 7900

LuxRender



Intel Xeon E5-2650 2.00GHz with 64GB, 2 NVidia Tesla M2075

Thank you