hmpp

Manycore
portable programming

Benefit from the performance of GPU accelerated systems while reducing your development efforts

Portability

- Protect your software investment

- Remain independent from the hardware platform

- Do not lock to a vendor specific API

- Work with C and Fortran standard compilers

Scalability

- Distribute computations between CPU and accelerators

- Provide hardware interoperability

- Complementary to OpenMP™ and MPI

Rapidly develop GPU accelerated applications with a source to source tool

■ A hybrid compiler with powerful CUDA - CAL/IL - OpenCL generators

With the HMPP™ target generators, instantaneously prototype and evaluate the performance of the hardware-accelerated critical functions.

The code generators are specifically designed to extract the most of data parallelism from your C and Fortran kernels and translate them into NVIDIA® CUDA™, AMD CAL/IL and/or OpenCL.

HMPP Workbench includes a C and Fortran compiler, code generators and a runtime that seamlessly integrate in your environment and make use of the NVIDIA CUDA, AMD CAL/IL or OpenCL development tools and drivers.
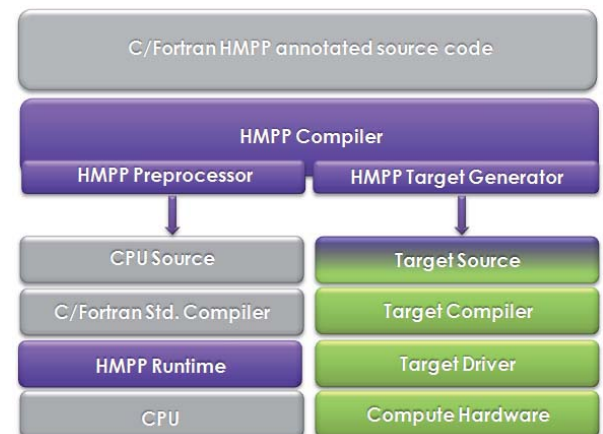
HMPP, the best way to master performance

■ HMPP directives: a high level abstraction for manycore programming

Based on a set of OpenMP™-like directives that preserve legacy codes, HMPP fully leverages the performance offered by most of today's stream processors and vector units.
HMPP offers a standardized interface between your scientific algorithm and fast evolving target code by insulating hardware specific implementation of functions from your legacy code.

Complementary to OpenMP and MPI, HMPP lets you develop parallel hybrid applications that mix the best of today's available parallel tools.

■ Dynamic application scaling

By providing different target versions of computations that are offloaded to the available hardware compute units, an HMPP application dynamically adapts its execution to multi-GPUs systems and platform configuration. This guarantees the scalability and interoperability of your application.

Developers

Rapidly integrate hardware accelerators in your application while preserving its portability.

OEMs

Provide software developers with standard programming tools that keep their applications interoperable with your evolutive hybrid platforms.

ISVs

Embedd HMPP in your product and offer a single software that leverages the computing power of all the different configurations of your customers manycore platforms.

A set of directives to develop manycore applications

In C and Fortran applications, the HMPP directives let you define and execute codelet functions to be offloaded in GPU accelerators.

In the main application, indicate the call sites of the codelets and their synchronous or asynchronous execution property.
By preloading data before the execution of the codelets and uploading the results whenever they are required in the main application, you optimize the use of the memory bandwidth.

Pipeline GPU computations and implement efficient CPU-GPU communication patterns that leverage the computation power of GPUs.

Use the asynchronous property of the directives to interlace data transfers and codelet execution. Build real hybrid computations that distribute their workload over the compute units.

```
#pragma hmpp sgemm codelet, args[m;n;k;alpha;beta;a;b].io=in, &
#pragma hmpp sgemm args[c].io=inout, target=CUDA:BROOK
void sgemm(int m, int n, int k,
                  float alpha, float a[m][k], float b[k][n],
                  float beta, float c[m][n]);

int main(int argc, char **argv) {

...

/* Allocate device and memory */
#pragma hmpp sgemm allocate, args[c].size={M,N}
/* Prefetch all data, alpha and beta are constant */
#pragma hmpp sgemm advancedload, args[alpha;beta;a;b;c], &
#pragma hmpp sgemm args[m;n;k;n;alpha;beta].const=true
...

#pragma hmpp sgemm callsite, args[alpha;beta;a;b;c].advancedload=true, &
#pragma hmpp sgemm asynchronous
  sgemm( M, N, K, alpha, t1, t2, beta, t3 );

/* asynchronous execution barrier */
#pragma hmpp sgemm synchronize
/* retrieve c output data   */
#pragma hmpp sgemm delegatedstore, args[c]
/* release the device */
#pragma hmpp sgemm release
...

  return 0;
}
```
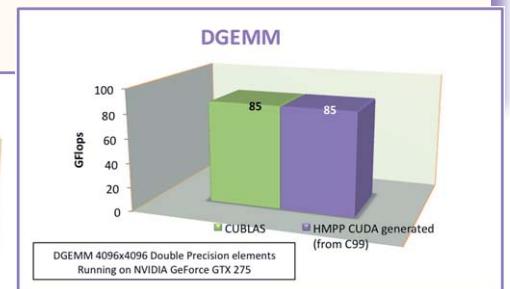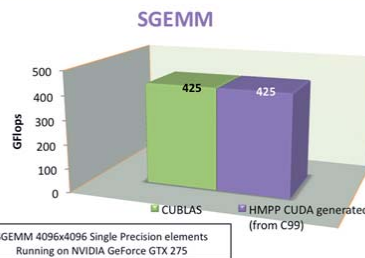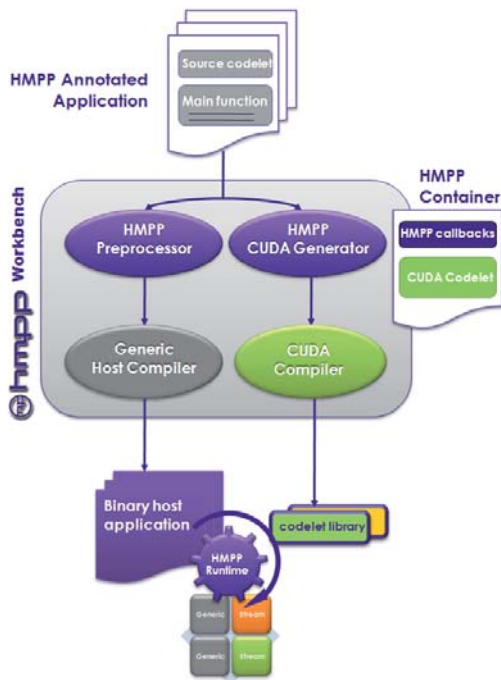

SGEMM 4096x4096 Single Precision elements Running on NVIDIA GeForce GTX 275

DGEMM 4096x4096 Double Precision elements Running on NVIDIA GeForce GTX 275

A single compilation command

From the HMPP annotated application, HMPP separately compiles the native host application and the GPU accelerated codelet functions as software plugins.

The codelets are translated in the NVIDIA CUDA and/or AMD CAL/IL by the HMPP Target Generator and compiled with the hardware vendor tools.
Linked with the HMPP Runtime, the native host application is able to execute stand-alone or to load and run the target codelet libraries when GPUs are present and available.

The HMPP Workbench seamlessly integrates with your standard optimizing compiler and the hardware vendor tools.

Supported platforms and compilers

GPUs
- All NVIDIA® Tesla™
- NVIDIA® CUDA™ compatible graphics products (GTX280, Quadro FX5800, GeForce 8800GTX, 9xx, ...)
- AMD FireStream™ 9170, 9250
- CAL compatible graphics products

Compilers
- GNU gcc 4.1 and above
- Intel icc version 9.1 and above
- Intel ifort version 9.1 and above

Operating systems

- Any x86_64 kernel 2.6 Linux distribution with libc coming with g++ 4.x and above.

- HMPP has been validated with some of the below Linux distributions:
    • Debian 4.0 and above
    • RedHat Entreprise Linux 5.x and above
    • OpenSuse 11.x and above
    • SLES 11.0
    • Ubuntu 8.10
    • ...
- Windows