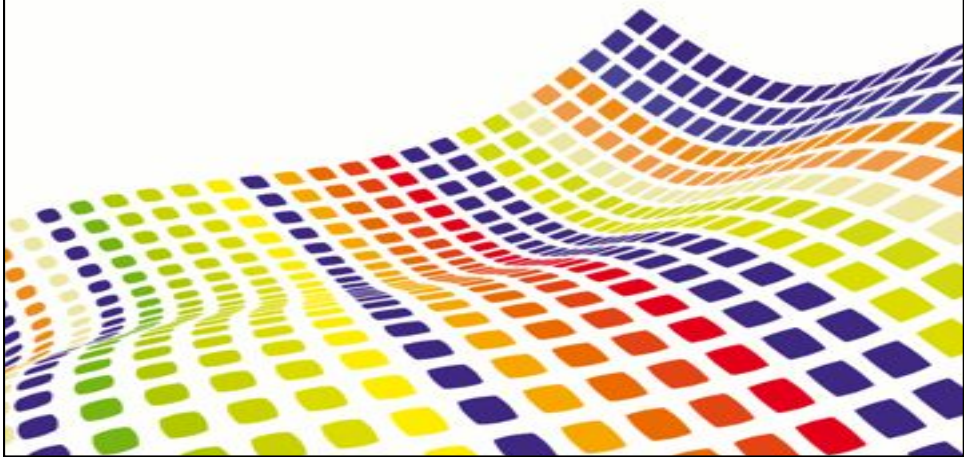
A Methodology for Code Migration and an Economical View of GPU Computing

CAPS



Free Lunch is Over, Codes Have to Migrate!

CAPS

- Many existing legacy codes needs to migrate to new architectures
  - mostly written in C/C++ and Fortran
- Computing power comes from parallelism
  - Hardware (frequency increase) to software (parallel codes) shift
  - Driven by energy consumption
    - heterogeneity is the source of efficiency
    - Few large fast OO cores combined with many smaller cores (e.g. APU)
- Very wide configuration space due to heterogeneity
  - Not specific to GPUs but to fat node systems
  - Requires hybrid programming (e.g. MPI + OpenMP / OpenCL) and looking for tradeoffs
    - number of nodes vs. the compute power of each node
  - Simpler if did not require specific code migration/tuning

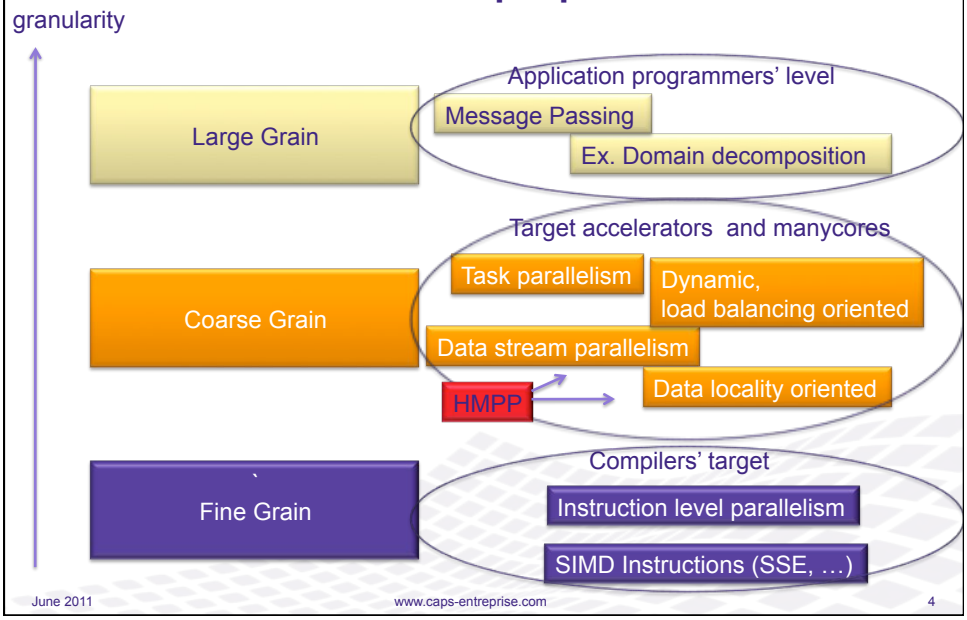
Legacy Codes Migration Challenges

CAPS

- Mastering migration cost
  - Ensuring an adequate return on investment
  - Minimizing risks as well as manpower
- Produce code that will last many architecture generations
  - It is safe to assume that the node architecture may change with the renewal of the computers
- Code that is still application developers friendly
  - App. developers may not be multicore / accelerator / parallelism savvy
  - Once ported, the application still needs to evolve
- Keeping a unique version, preferably mono-language, of the codes
  - Reduce maintenance cost
- Library use
  - No one-to-one replacement (e.g. FFT libraries)
  - Must interact with non library accelerated kernels

Heterogeneous hardware, multiple parallelism forms

CAPS



What is HMPP? (Hybrid Manycore Parallel Programming)

CAPS

- A directive-based multi-language (C and Fortran) programming environment
  - Help keeping software independent from hardware targets
  - Provide an incremental tool to exploit GPUs in legacy applications
  - Avoid exit cost, can be future-proof solution
- HMPP provides
  - Code generators from C and Fortran to OpenCL / CUDA
  - A compiler driver that handles all low level details of GPU compilers
    - Code generation and data transfer directives
    - GPU code optimization directives
  - A runtime to allocate and manage GPU resources
- Source to source compiler
  - CPU code does not require compiler change
  - Complement existing parallel APIs (OpenMP or MPI)

HMPP Basic Directives

CAPS

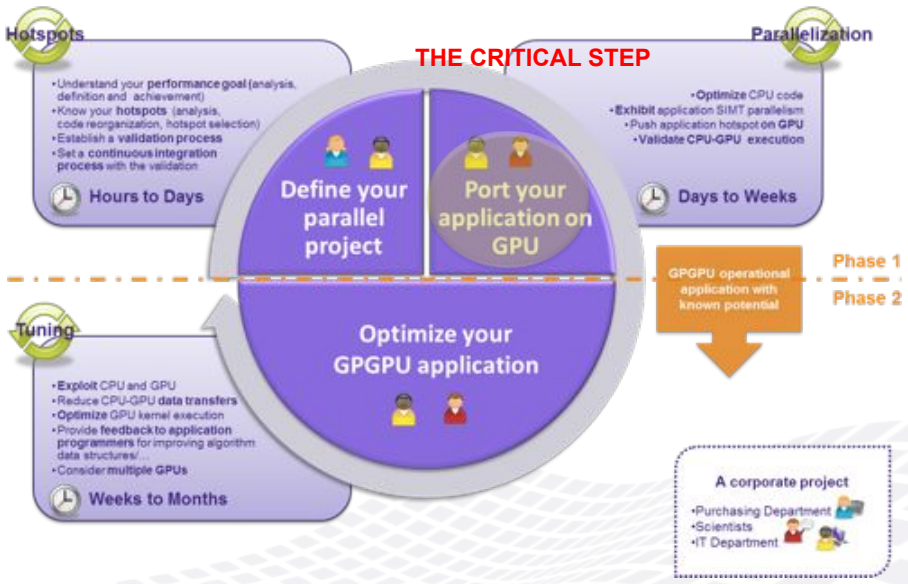
- The `Codelet` directive indicates the function to compile for GPU
- The `callsite` directive performs a Remote Procedure Call (RPC) onto the GPU

```
#pragma hmpp call1 codelet, target=OpenCL
void myFunc(int n, int A[n], int B[n]){
    int i;
    for (i=0 ; i<n ; ++i)
        A[i] = A[i]+B[i]+1;
}

void main(void)
{
    int [100][10000], Y[10000];
    ...
    for (i=0;i<100;i++){
      #pragma hmpp call1 callsite,  …
      myFunc(10000, X[i], Y);
    }
    ...
}
```

Methodology to port applications

CAPS



Go / No Go

CAPS

Go

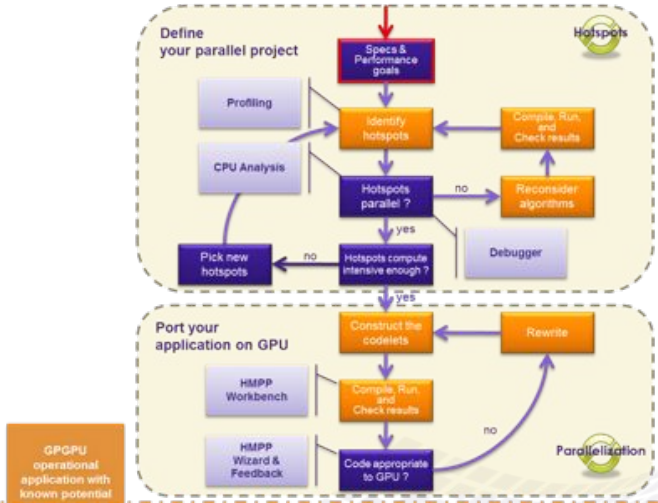
- Dense hotspot
- Fast kernels
- Low CPU-GPU data transfers
- Prepare to manycore parallelism

No Go

- Flat profile
- Slow GPU kernels (i.e. no speedup to be expected)
- Binary exact CPU-GPU results (cannot validate execution)
- Memory space needed

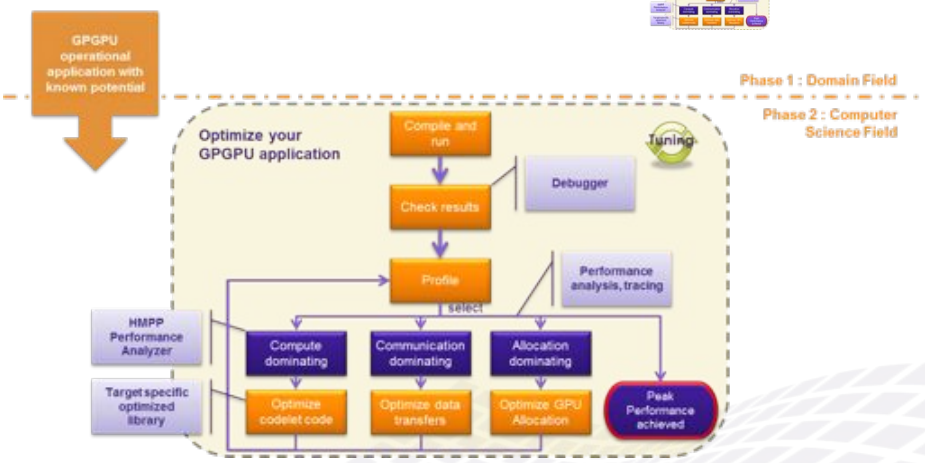
Phase 1 (details)

CAPS



Phase 2 (details)

CAPS



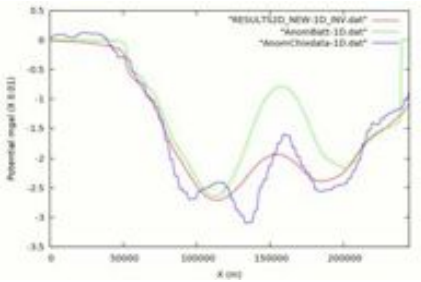
Geophysics

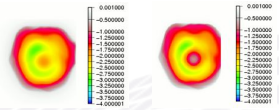
CAPS

GENCI

3D Poisson equation conjugate gradient

- Resource spent
  - 2 man-month
- Size
  - 2kLoC of F90 (DP)
- CPU improvement
  - **X1,73**
- GPU C1060 improvement
  - x 5,15 over serial code on Nehalem
- Main porting operation
  - highly optimizing kernels





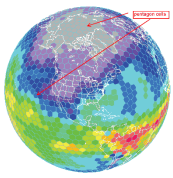
The finite volumes method (left) is more accurate than the analytic solution (right) which over estimates the central peak

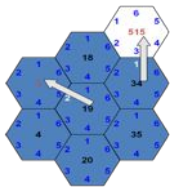
Weather Forecasting

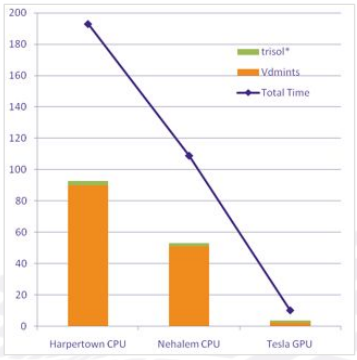
CAPS

A global cloud resolving model

- Resource spent
  - 1 man-month (part of the code already ported)
- GPU C1060 improvement
  - 11x over serial code on Nehalem
- Main porting operation
  - reduction of CPU-GPU transfers
- Main difficulty
  - GPU memory size is the limiting factor







Computer vision & Medical imaging

CAPS

TOHOKU UNIVERSITY

Tōtoku C-tec

MultiView Stereo

- Resource spent
  - 1 man-month
- Size
  - ~1kLoC of C99 (DP)
- CPU Improvement
  - x 4,86
- GPU C2050 improvement
  - x 120 over serial code on Nehalem
- Main porting operation
  - Rethinking algorithm

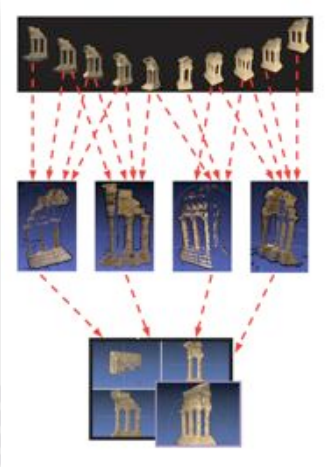


Image processing

CAPS

Edge detection algorithm

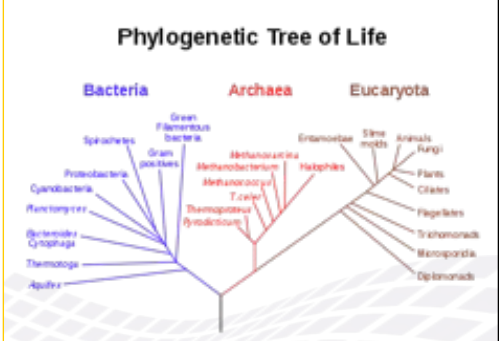
- Sobel Filter benchmark
- Size
  - ~ 200 lines of C code
- GPU C2070 improvement
  - x 25,8 over serial code on Nehalem
- Main porting operation
  - Use of basic HMPP directives

Biosciences, phylogenetics

CAPS

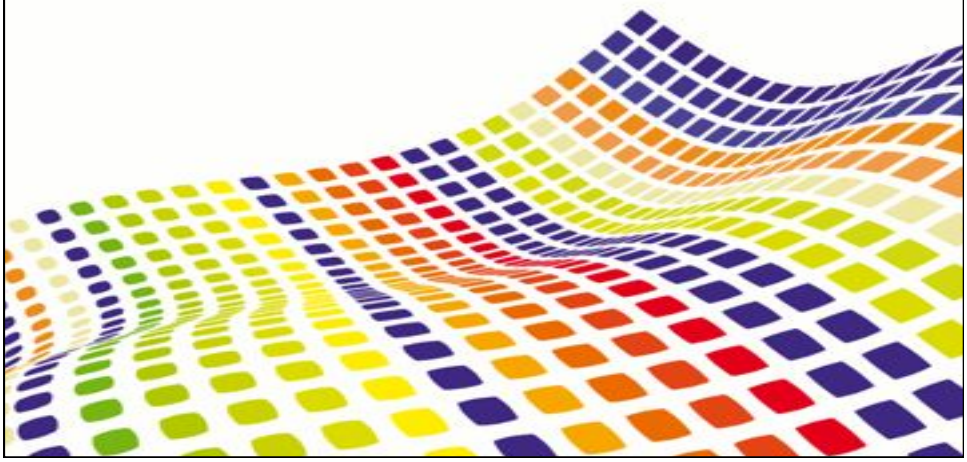
Phylip, DNA distance

- In association with the HMPP Center Of Excellence for APAC
- Computes a matric of distances between DNA distances
- Resource spent
  - A first CUDA version developed by Shanghai Jiao Tong University, HPC Lab
  - 1 man-month
- Size
  - 8700 lines of C code, one main kernel (99% of the execution time)
- GPU C2070 improvement
  - x 44 over serial code on Nehalem
- Main porting operation
  - Kernel parallelism & data transfer coalescing leverage
  - Conversion from double precision to simple precision computation



CAPS

An Economical View of GPU Computing



CAPEX-OPEX Analysis for a Heterogeneous System

CAPS

- Capital Expenses (CapEx)
  - System acquisition cost
  - Software migration cost
  - Software acquisition cost
  - Teaching cost
  - Real estate cost
- Operational Expenses (OpEx)
  - Energy cost (system + cooling)
  - Maintenance cost
- For a given amount of compute work, the CapEx-Opex analysis indicates the “real” value of a given system
  - For instance, if I add GPU do I save money? And how many should I add?
  - Then should I use slower CPU?

Application Speedup and CapEx-OpEx

CAPS

- Adding GPUs/accelerators to the system
  - Increases system cost
  - Increases base energy consumption (one GPU = x10 watt idle)
- Exploiting the GPUs/accelerators
  - Decreases execution time, so potentially the energy consumption for a given amount of work
  - Reduces the number of nodes of the architecture
    - Threshold effect on the number of routers etc.
  - Requires to migrate the code
- Multiple views of the value of application speedup
  - Shorten time-to-market
    - Threshold effect
  - More work performed during the lifetime of the system

CapEx Hardware Parameters

CAPS

- Choice of the hardware configuration can be:
  - Fast CPU + Fast GPU (expensive node)
  - Slow CPU + Fast GPU
  - Fast CPU + Slow GPU
  - Slow CPU + Slow GPU
  - Fast CPU
  - Slow CPU

Small systems:
- a few nodes (1-8)
- cost x10k€

Large systems
- many nodes (x100)
- cost x1M€

- Nodes performance impact on the number of nodes
  - More nodes means more network with non negligible cost and energy consumption
  - Less nodes may limit scalability issues if any
- Application workload analysis is the only way to decide
  - Optimizing software can significantly increase performance and so reduce needed hardware
  - Code migration to GPU is on the critical path

CapEx: Code Migration Cost

Mastering the cost of migration has a significant impact on the total cost for small systems

Typical effort:
- Manpower: a few Man-Months
- Cost: x 10k€

- Migration cost
  - Learning cost
  - Software environment cost
  - Porting cost
- Migration cost is mostly hardware size independent
  - Not an issue for dedicated large systems
  - Different if the machine aims at serving a large community
- Main migration benefit is to highlight manycore parallelism
  - Not specific to one kind of device
  - Implementation is specific
- Constructor specific implementation solution
  - Amortize period similar to the one of the hardware (3 years)
- Agnostic parallelism expression
  - Using portable solution for multiple hardware generations (amortized on 10 years)
  - Of course not that simple! Still requires some level of tuning
- May be very useful for non scalable message passing code

Two Applications Examples

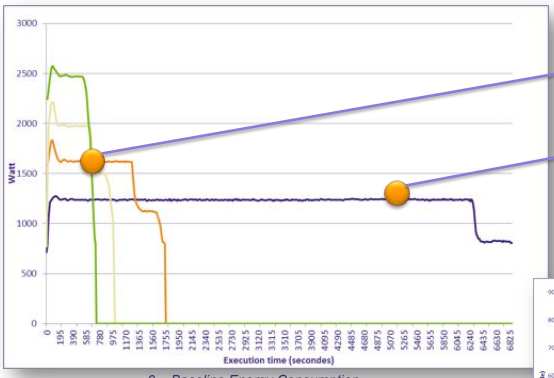
Application 1

- Field: Monte Carlo simulation for thermal radiation
- MPI code
- Migration cost: 1 man month

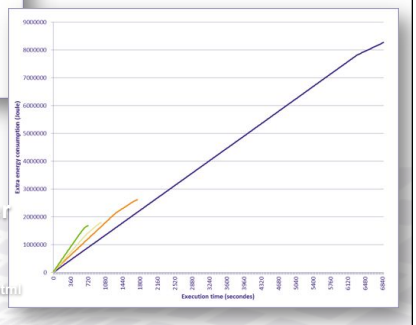
Application 2

- Field: astrophysics, hydrodynamic
- MPI code
- Requires 3 GPUs per node for having enough memory space
- Migration cost: 2 man month

Power Consumption Application 1



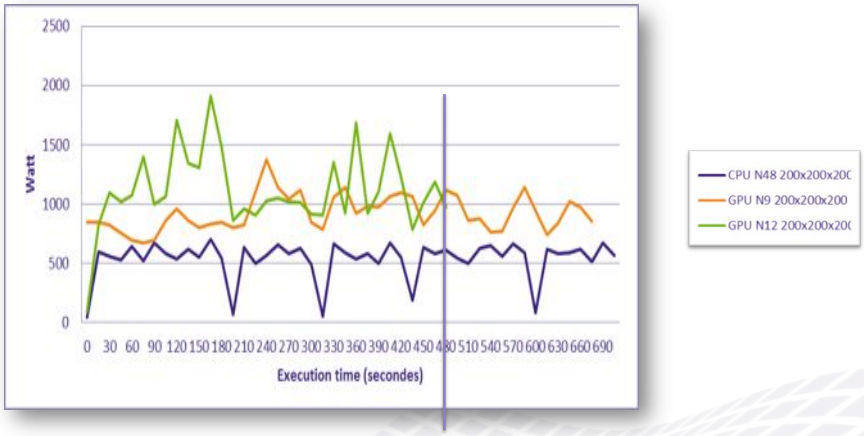
0 = Baseline Energy Consumption



Power usage effectiveness (PUE) =
Total facility power / IT equipment power

Current 1.9, best practice 1.3
Src: http://www.google.com/corporate/datacenter/efficiency-measurements.html

Power Consumption Application 2

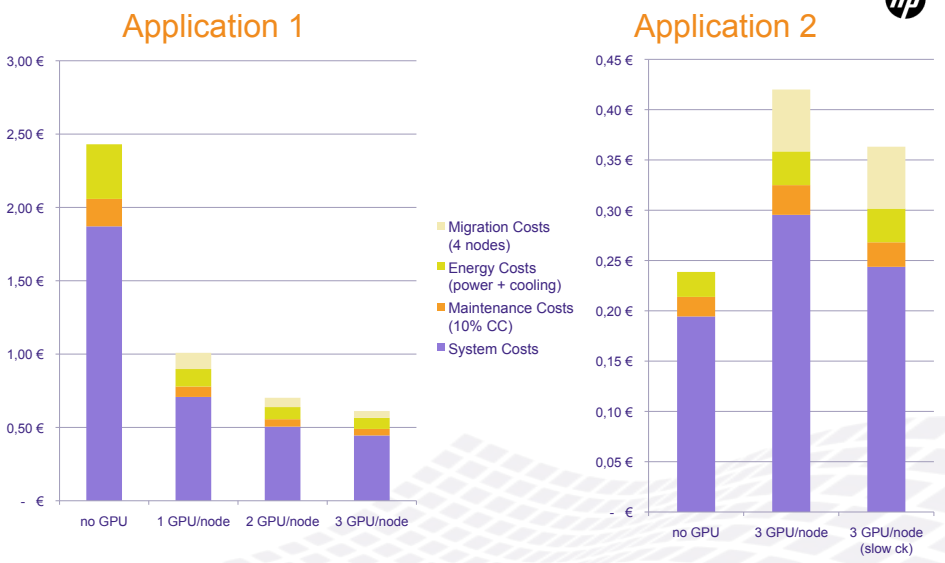


CAPEX-OPEX Overview

- Comparison on an equivalent workload
  - CAPEX = System costs + Migration costs
  - OPEX = Energy cost + Computer maintenance cost (10% Computer costs)

Configuration	Execution time (s)	System Costs	Maintenance Costs	Energy Costs	CAPEX +OPEX
Application 1		Migration cost = 1 man.month			
4 nodes	6862	1.87€	0.19€	0.37€	**2.43€**
4 nodes + 4 GPUs	1744	0.71€	0.07€	0,12€	**0.90€**
4 nodes + 8 GPUs	1000	0.51€	0.05€	0,08€	**0.64€**
4 nodes + 12 GPUs	731	0.45€	0.04€	0,08€	**0.57€**
Application 2		Migration cost = 2 man.month			
4 nodes	713	0.19€	0.02€	0.025€	**0.239€**
4 nodes + 12 GPUs	485	0.30€	0.03€	0.034€	**0.358€**
4 nodes (slow ck)+ 12 GPUs	500 (estim.)	0.24€	0.02€	0.034€	**0.302€**

Cost per Run



Conclusion

- Heterogeneous architectures are becoming ubiquitous
  - In HPC centers but not only
  - Tremendous opportunities but not always easy to seize
  - CPU and GPU have to be used simultaneously
- Legacy codes still need to be ported
  - Software migration required understanding options
    - Do not want to backtrack
  - A methodology supporting tools is needed and must provide a set of consistent views
  - The legacy style is not helping
  - Highlighted parallelism for GPU is useful for future manycores
- HMPP based programming
  - Helps implementing incremental strategies
  - Is being complemented by a set of tools (DevDeck™)

Perspectives

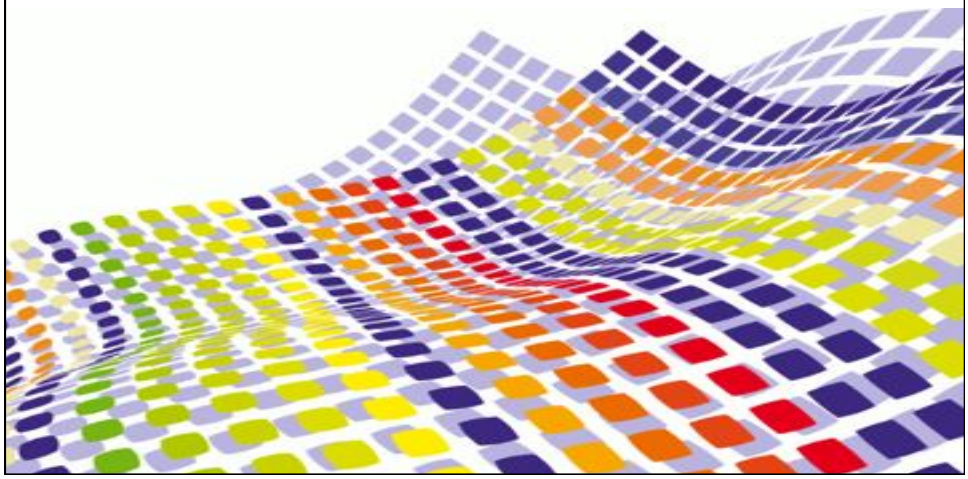
CAPS

- Need for new standard programming
  - OpenHMPP initiative launch by CAPS
  - http://www.openhmpp.org/
- Energy consumption control at software level
  - Is energy saving cost worthwhile the software tuning cost?
- Cloud technology
  - All manycore issues and more …

HMPP for Future Manycores

CAPS

- Current HMPP (2.x)
  - Agnostic directive-based style
  - Target stream parallelism on accelerators
  - High level expression of stream oriented parallelism
  - Mostly deals with one GPU per threads, no GPU sharing
  - Oriented toward performance and device memory saving
- Next HMPP (3.x)
  - Agnostic directive-based with more API for expert programmers
    - Adaptive programming
  - High level expression of stream oriented parallelism
  - Target accelerators and CPU cores
  - Handles multiple GPUs, data distribution
  - Easier handling of data management between CPU and GPU

CAPS

http://www.caps-entreprise.com
http://twitter.com/CAPSentreprise
http://www.openhmpp.org